Top Martin Marietta

MCR-70-365

DEVELOPMENT OF A TEST AND FLIGHT ENGINEERING ORIENTED LANGUAGE

L 2

PHASE II REPORT

W. F. Kamsler
C. W. Case
E. L. Kinney
J. Gyure

Martin Marietta Corporation
Denver Division
P. O. Box 179
Denver, Colorado 80201

October 1970

Interim Report for September - October 1970

Prepared for
National Aeronautics and Space Administration
John F. Kennedy Space Center

Sqt-65209

NOTICE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Test and Flight Engineer Oriented Language		5. Report Date October 9, 1970 6. Performing Organization Code
7. Author(s) W. F. Kamsler, C. W. Case, J. Gyure, E. L. Kinney		8. Performing Organization Report No. MCR-70-365
9. Performing Organization Name and Address MARTIN MARIETTA CORPORATION Denver Division P. O. Box 179 Denver, Colorado 80201		10. Work Unit No. 11. Contract or Grant No. NAS10-7308
12. Sponsoring Agency Name and Address National Aeronautics and Space Adminstration Kennedy Space Center Florida 32899		13. Type of Report and Period Covered Phase II Report September 1970 to Oct. 1970 14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

This report performs an analysis of the Space Shuttle system and details those requirements which impact on the development of a test and flight engineer language.

The necessary resulting language characteristics are analyzed and reasons are given as to why they are necessary in the proposed test oriented language.

17. Key Words	18. Distribution Statement
Test Oriented Language Space Shuttle Computer Controlled Test Equipment	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
UNCLASSIFIED	UNCLASSIFIED	90	

PREFACE

This report contains the results of the Phase II effort in the development of a Test and Flight Engineer Oriented language.

An examination of the proposed Space Shuttle program was made to determine how it and its support equipment might influence the language design.

The general characteristics and capabilities of the new language are selected and described, along with their justifications.

CONTENTS

1.0 INTRODUCTION

This report covers the results of the Phase II study task related to the development of a Test and Flight Engineer Oriented Language.

The task included a brief study of Space Shuttle and its support equipment. From this effort the problems and requirements of readying a Space Shuttle for launch and operation were determined. It must be realized that the Space Shuttle is now in a Phase B design study. It is expected that changes in design will be made but that these changes will have minimal effect on the test and checkout requirements.

The requirements of the user(s) were investigated because the language provides his interface to the test and control activities. These considerations influence the design of the Test and Flight Engineer Oriented language.

Further work was done on the selection of characteristics for the language. This effort took into consideration the work done in Phase I on the study of previously developed test oriented languages.

The problems of onboard checkout, maximum autonomy, and rapid turn around time were investigated taking into consideration ground support equipment and vehicle verification prior to launch. Suggestions are made on how this can be implemented with minor impact on the Space Shuttle ground rules.

The need for concurrent testing (execution of multiple test procedures simultaneously) was considered and found to be desireable.

Readability of the test language was deemed to be of prime importance. Provisions will be included to speed up the writing of test programs through the use of abbreviations and conventions; however, the test (computer prepared) printouts will completely define the test actions in English-like statements.

Other defined language characteristics will assure acceptance by all involved with the Space Shuttle program.

Phase III will select terminology and provide a specification for the suggested Test and Flight Engineer Oriented Language.

2.0 SPACE SHUTTLE CONFIGURATION

A brief description of the Space Shuttle is presented to typically define the vehicle and show the nature of the equipment and subsystems that will influence development of the Test and Flight Engineer Oriented Computer language. At this time the design of the Space Shuttle is being defined in Phase B Studies; therefore this information is subject to change. Alternative configuration have been considered, and would have insignificant impact on language requirements.

2.1 Ground Rules.

The following ground rules apply:

- Two stage reuseable vehicle.
- Maximum onboard autonomy.
- Vehicle capability of 100 mission cycles.
- High launch rate (25 to 75 per year).
- Self-sustaining orbital lifetime of 7 days extendable to 30 days with additional expendables from the payload.
- 14 day ground turn-around time including inspection, refurbishment, replacement, and retest.
- No inflight maintenance.
- Mechanical subsystems shall fail operational (1st failure) fail safe (2nd failure).
- Electronic subsystems shall fail operational (1st failure) fail operational (2nd failure) and fail safe (3rd failure).
- Airline type operations; two man crew with one man capability, shirt sleeve environment.
- Retest requirements involve only the replaced LRU.
- Technology baseline 1972.

From these guidelines several teams are developing conceptual Space Shuttles. These designs feature fully integrated redundant avionic systems controlled via data buses from centralized multiple digital computers.

This section of the report will emphasize the avionic systems controlled and monitored by the central computer system.

2.2 Booster System Functions

The baseline Booster Avionics System provides the following functions:

1. Engine Control - Jet and Rocket
2. Voice Communication to Ground and Orbiter
3. Attitude Reference
4. Flight Control - Aerodynamic and Reaction Jets - Manual and Automatic
5. Navigation from Lift off to Landing.
6. Displays for Pilot and Copilot
7. Power Control and Conditioning

8. Control, Checkout and Status Monitoring of Vehicle Subsystems

   - ECLS
   - Hydraulics
   - Landing Gear
   - Lights
   - Hatches and Doors
   - Electrical Power
   - Propellant
   - Separation

9. Flight Recording; Maintenance and Flight Data
10. Computer Executive and Data Bus Control
11. Onboard Checkout, Ground Checkout
12. Mission Planning

The baseline configuration of the Booster Avionics System implements the concept of quad-redundant subsystems for safety functions and triply redundant systems for mission success items. This concept meets the fail operational, fail operational, fail safe criteria proposed by NASA. Convenience items are in quantities of one or two. The equipment is generally arranged such that redundant systems are physically separated in the vehicle, thus preventing loss of capability in the event of localized damage such as a spill, explosion or collision.

Each of the subsystems is controlled through multiplexed data buses from central computers. Four data buses run forward and aft from the central computers. The data buses interface with the avionics equipments via (semi) standardized interface units (IU).

Peripheral data processing and formatting is utilized to simplify and reduce data rates for the interfaces between the data distribution system and the Inertial Reference Units, the Crew Displays and the Jet and Main Engines.

Electric power is generated by redundant turbo-alternators and distributed through four power buses. The power buses are physically located in the same manner as data buses to maximize the probability of surviving an incident which would disable some portion of the system. Power switching between the turbo-alternators and power buses is implemented by manual/hardwire operation as is the power switching to the four central computers. All other switching of power to subsystem LRU's is under computer control via the data bus.

Selection of active LRUs will be accomplished through the use of subsystem BITE, computer self-tests, and voting in the central computers. After the second subsystem failure of a quad-redundant subsystem, voting will be discarded and active LRU selections will depend primarily on BITE and crew decisions.

Selected data for maintenance and trend analysis will be recorded in flight. This will include BITE status, dissenting votes, etc., and such data as zero g accelerometer bias and vacuum engine thrust measurements. Data which can be obtained during ground checkout may not be recorded in flight.

A typical configuration of the Booster Avionics System is shown in Figures 2-1 through 2-7.

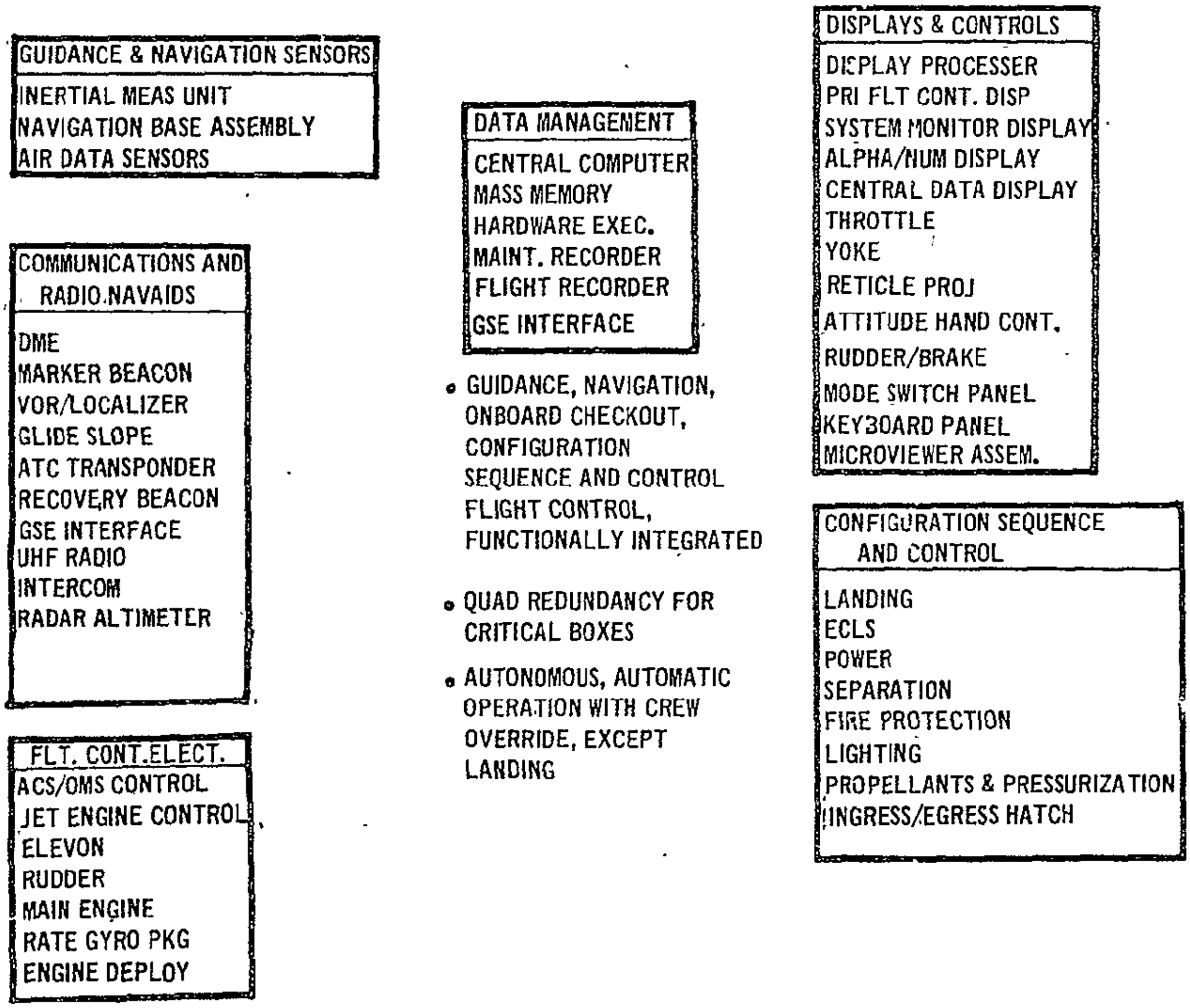


Figure 2-1 Booster Integrated Avionics System Baseline Configuration

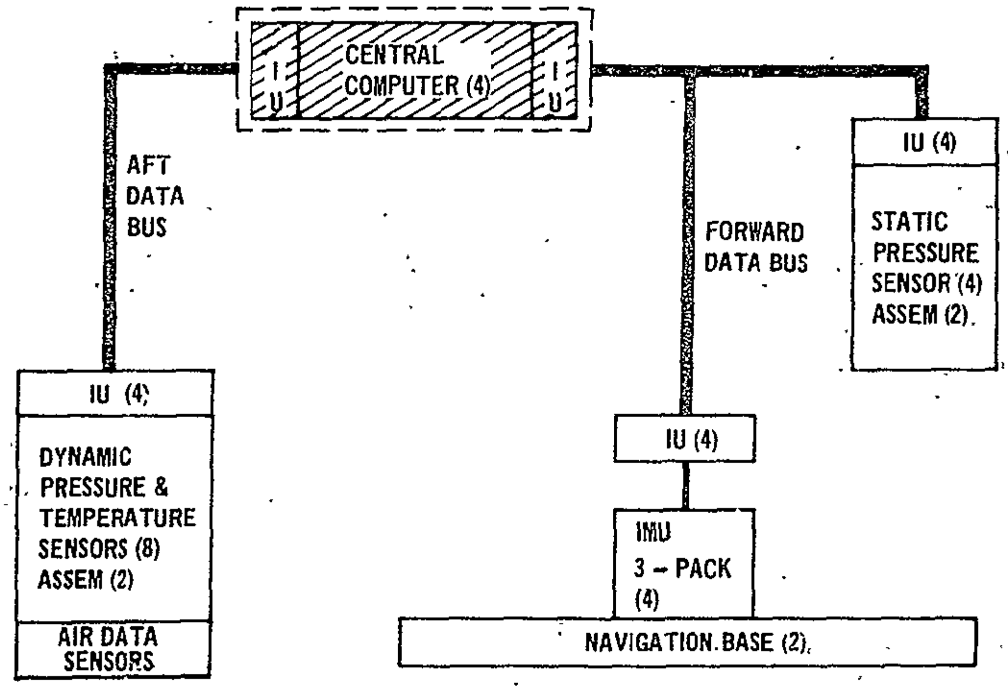


Figure 2-2 Booster Guidance and Navigation Sensors Subsystem

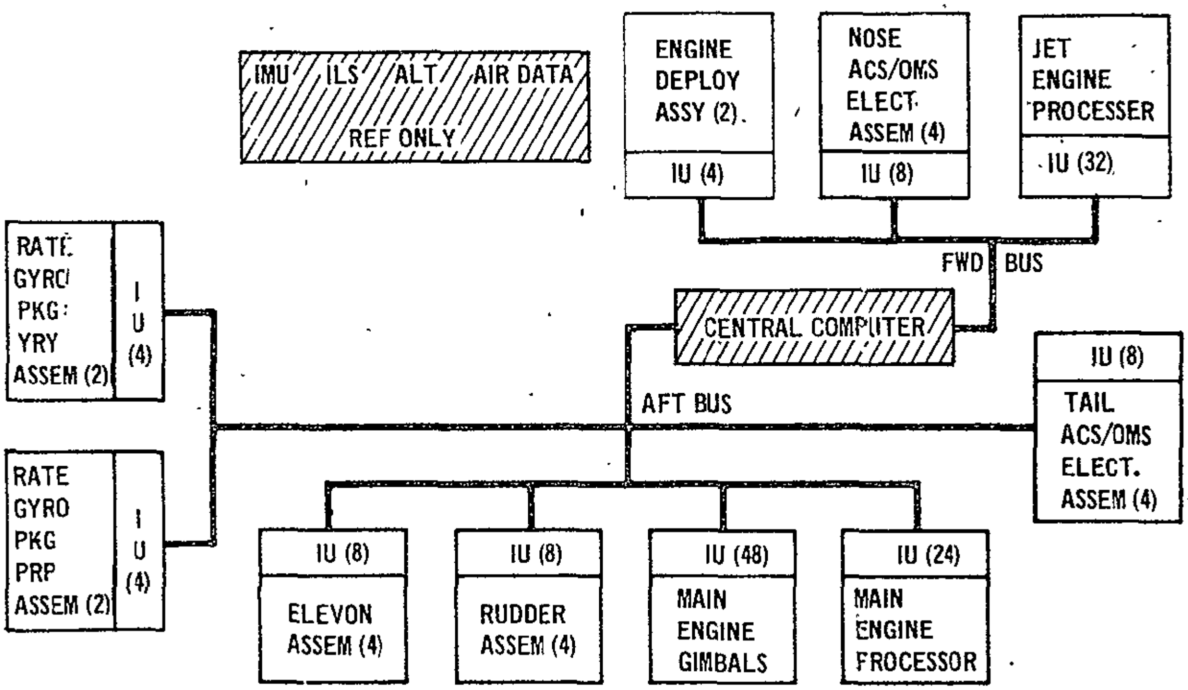
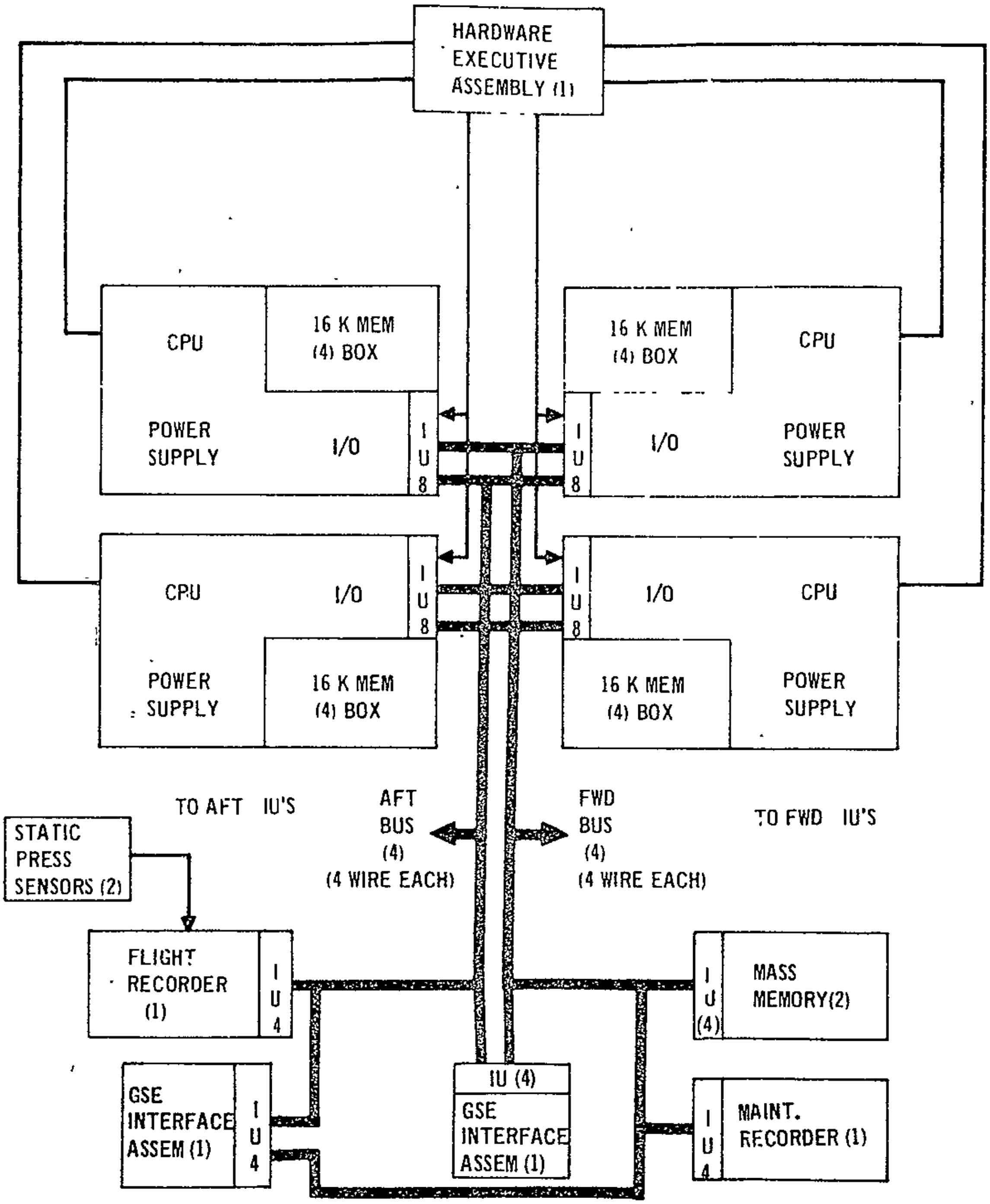


Figure 2-3 Booster Flight Control Electronics



- ONBOARD CHECKOUT, CONFIG. SEQ, FLIGHT CONT, G&N COMPUTATION ALL INTEGRATED
- FOUR UNIPROCESSORS, ONE IN CONTROL, AUTOSWITCH BY EXEC UPON FAILURE.
- STANDARD DIGITAL INTERFACES TO ALL LRU'S VIA DATA BUS

Figure 2-4 Booster Data Management Subsystem

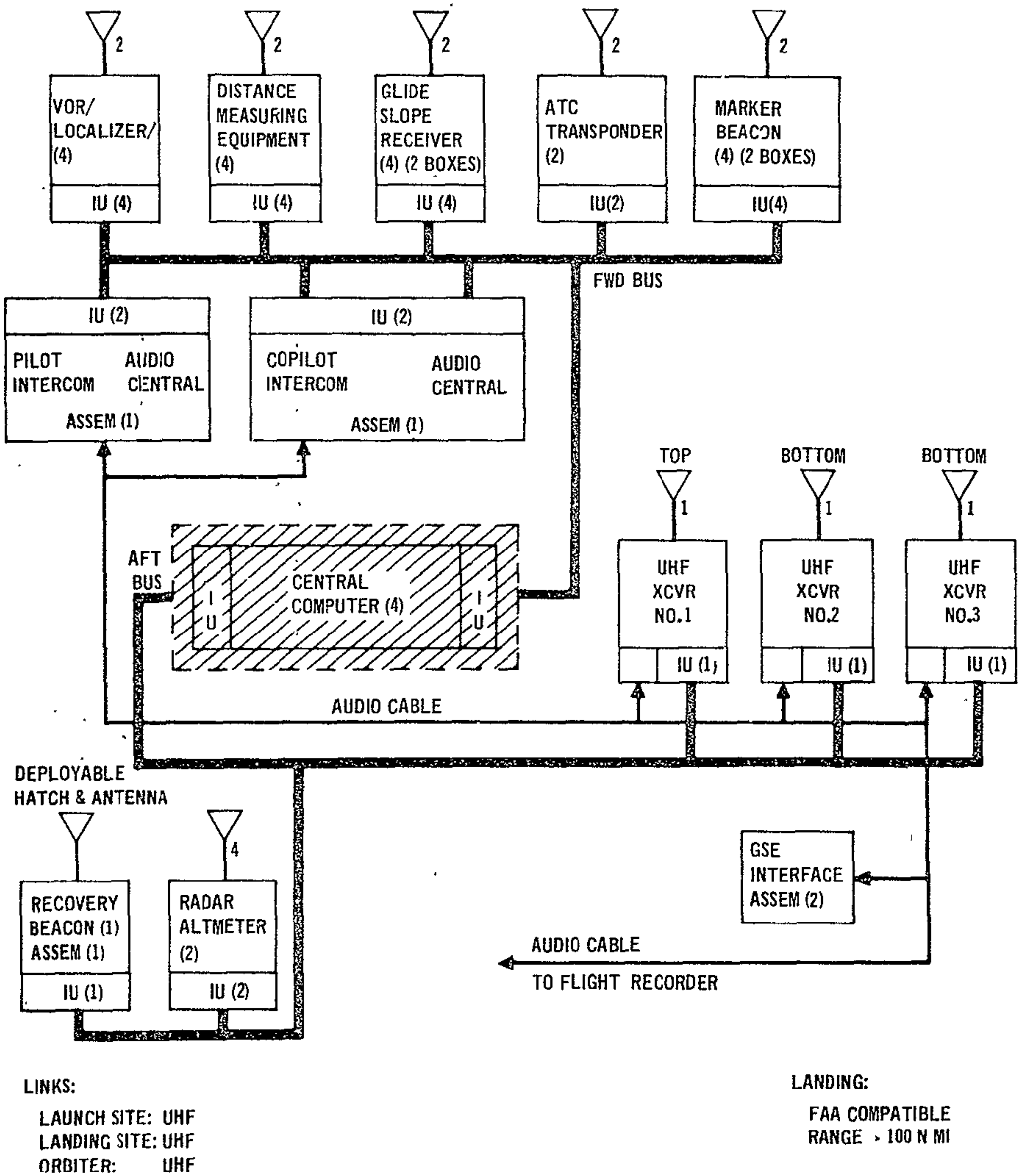


Figure 2-5 Booster Communications and Navaids Subsystem

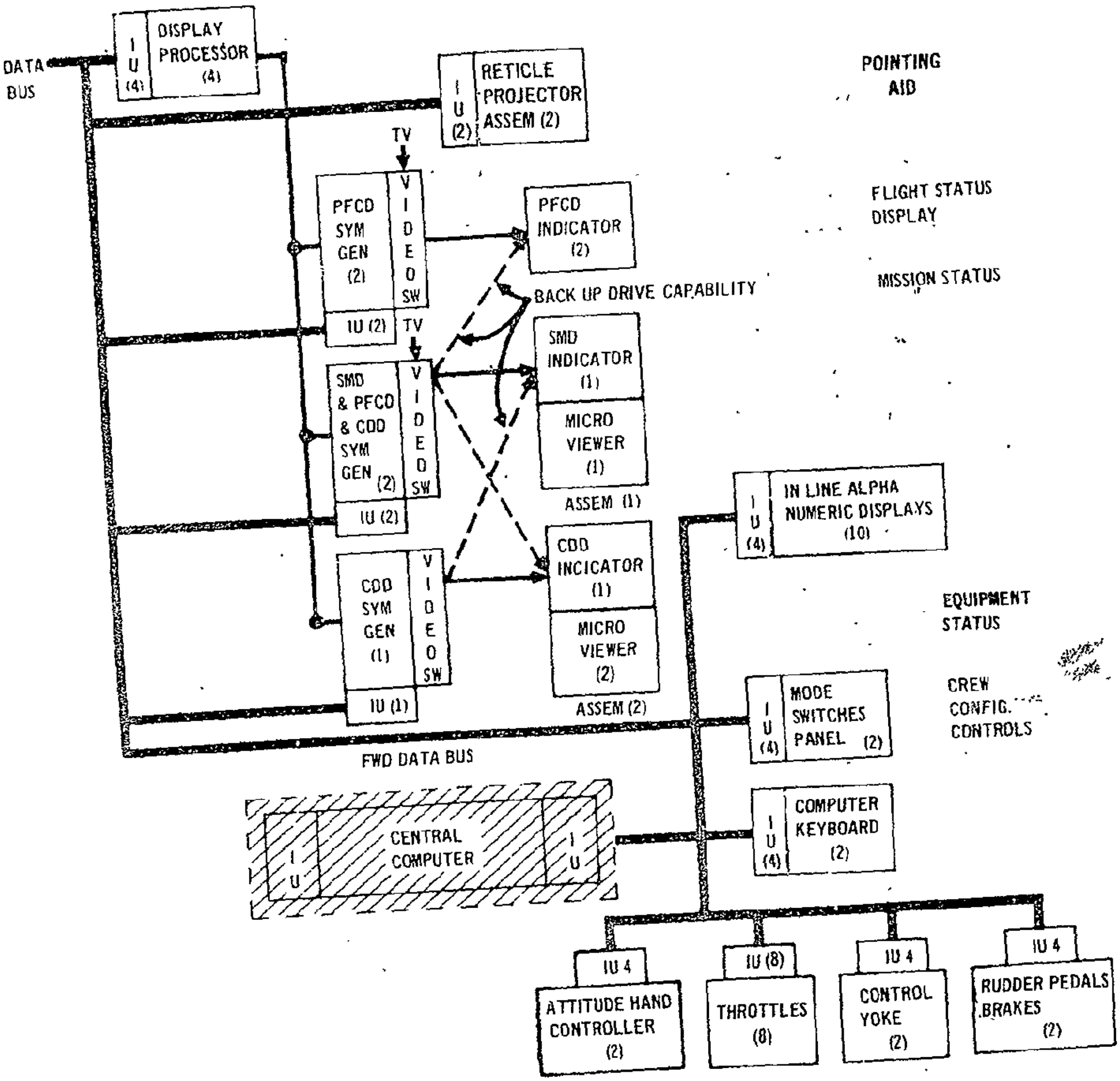
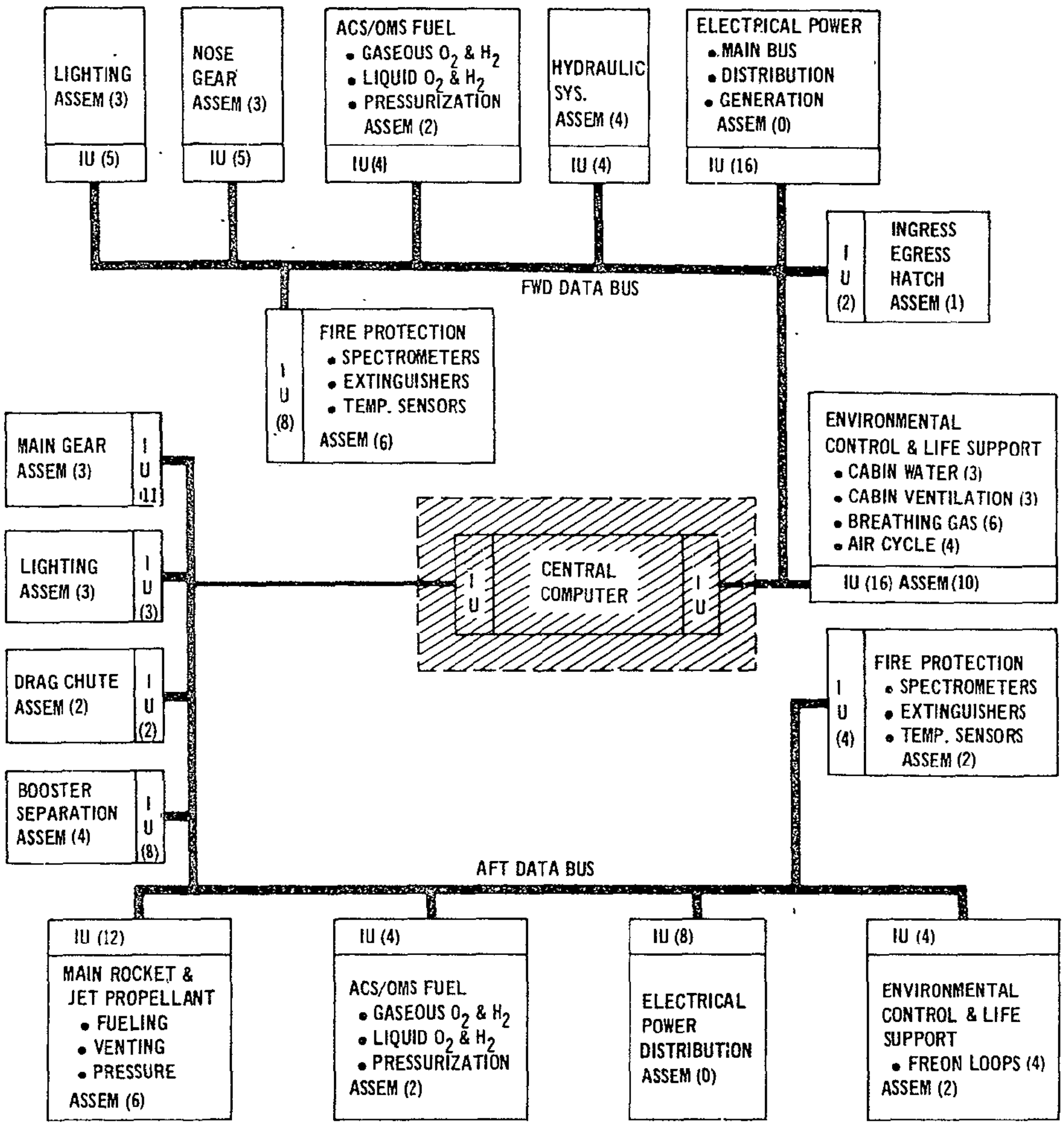


Figure 2-[illegible] Booster Controls and Displays Subsystem



- COMPLETE INSTRUMENTATION PROCESSING AND STATUS MONITORING
- TOTAL COMMAND CONTROL USING DMS AND DATA BUS

Figure 2-7 Booster Configuration and Sequence Control Subsystem

2.3 Orbiter System Functions

The baseline Orbiter Avionics System provides the following functions:

1. Engine Control - Jet and Rocket
2. Communication to Ground, Booster and Space Station - Voice and Data
3. Attitude Reference
4. Flight Control - Aerodynamic and Reaction Jets - Manual and Automatic
5. Navigation in Orbit and Atmosphere
6. Guidance During Boost, Orbital Burns, Entry and Landing
7. Displays for Pilot and Copilot
8. Power Control and Conditioning
9. Control, Checkout and Status Monitoring of Vehicle Subsystems

   - ECLS
   - Hydraulics
   - Landing Gear
   - Lights
   - Hatches and Doors
   - Fire Protection
   - Electrical Power
   - Propellant
   - Separation
   - Payload

10. Flight Recording; Maintenance and Flight Data
11. Computer Executive and Data Bus Control
12. Onboard Checkout, Ground Checkout
13. Mission Planning
14. Alignment of TV Cameras

The baseline configuration of the Orbiter Avionics System implements the concept of quad-redundant subsystems for safety functions and triply redundant systems for mission success items. This concept meets the fail operational, fail operational, fail safe criteria proposed by NASA. Convenience items are in quantities of one or two. The equipment is generally arranged such that redundant systems are physically separated in the vehicle, thus preventing loss of capability in the event of localized damage such as a spill, explosion or collision.

Each of the subsystems is controlled through one or more multiplexed data buses from a central computer located near the center of the vehicle. Peripheral data processing and formatting is utilized to simplify and standardize the interfaces between the data distribution system and the Inertial Reference Units, the Crew Displays and the Jet and Main Engines.

Electric power is generated by four fuel cells and distributed through four power buses. The power buses are physically located in the same manner as data buses to maximize the probability of surviving an incident which would disable some portion of the system. Power switching between the fuel cells and power buses is implemented by manual/hardware operation as is the power switching to the four central computers. All other switching of power to

subsystem LRU's is under computer control via the data bus.

Selection of active LRUs will be accomplished through the use of subsystem BITE computer self-test, and voting in the central computer. After the second subsystem failure of any quad-redundant subsystem, voting will be discarded and active LRU selections will depend primarily on BITE and crew decisions.

Selected data for maintenance and trend analysis will be recorded in flight. This will include BITE status, dissenting votes, etc., and such data as zero g accelerometer bias and engine thrust measurements.

A typical configuration of the Orbiter Avionics System is shown in Figures 2-8 through 2-14.

FLT CONT. ELECT

ACS/OMS CONTROL
JET ENGINE CONTROL
RATE GYRO PKG
AILERON
WING FOLD
ELEVATOR
FLAP
RUDDER
MAIN ENGINE

GUIDANCE & NAVIGATION

INERTIAL MEAS. UNIT
LASER/STAR TRACKER
HORIZON SCANNER
TRAINABLE TV
CCTV
NAVIGATION BASE ASSEM.
AIR DATA SENSORS
REND. RADAR*INTERFACE

*RENDEZVOUS RADAR ITSELF PART OF PAYLOAD WHEN REQUIRED

DATA MANAGEMENT

CENT. COMPUTER
MASS MEMORY
HARDWARE EXEC.
MAINT. RECORDER
FLT. RECORDER
PRINTER
GSE INTERFACE

DISPLAYS AND CONTROLS

DISPLAY PROCESSER
PRI FLT CONT DISPLAY
SYSTEM MONITOR DISPLAY
ALPHA/NUM. DISPLAY
CENTRAL DATA DISPLAY
THROTTLE
YOKE
TRANSLATE HAND CONT.
ATTITUDE HAND CONT.
SENS. POINT CONTROL
RUDDER/BRAKE
MODE SWITCH PANEL
KEYBOARD PANEL
MICROVIEWER ASSEM.
RETICLE PROJECTOR

COMMUNICATIONS & NAVAIDS

TRANSIT RECEIVER
DME
MARKER BEACON
VOR/LOCALIZER
GLIDE SCOPE REC.
ATC TRANSPONDER
RECOVERY BEACON
SHF RADIO
UHF RADIO
INTERCOM
SHF DISH ASSEMBLY
GSE INTERFACE
SPACE STATION INTERFACE
RADAR ALTIMETER
DIPLEXERS

- GUIDANCE, NAVIGATION ON BOARD CHECKOUT, CONFIGURATION SEQUENCE, & CONTROL, FLIGHT CONTROL FUNCTIONALLY INTEGRATED
- QUAD REDUNDANCY FOR CRITICAL BOXES
- AUTONOMOUS, AUTOMATIC OPERATION WITH CREW OVERRIDE, EXCEPT LANDING

CONFIGURATION SEQUENCE AND CONTROL

LANDING
ECLS
POWER
DOCKING LATCH
PAYLOAD DOORS
FIRE PROTECTION
LIGHTING
PROPELLANT & PRESSURIZATION
HATCHES

Figure 2-8 Orbiter Integrated Avionics System Baseline - Configuration

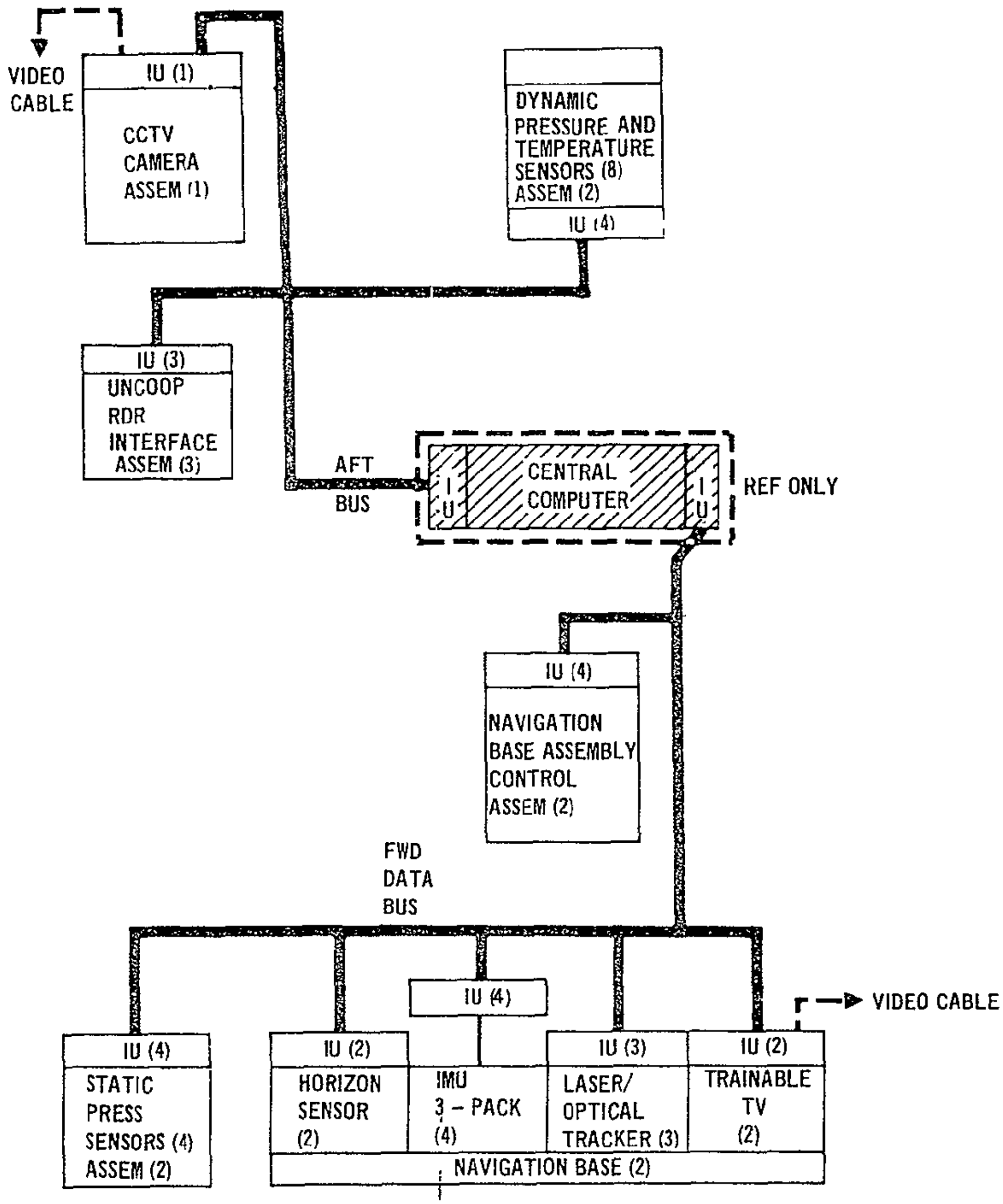


Figure 2-9 Orbiter Guidance and Navigation Subsystem

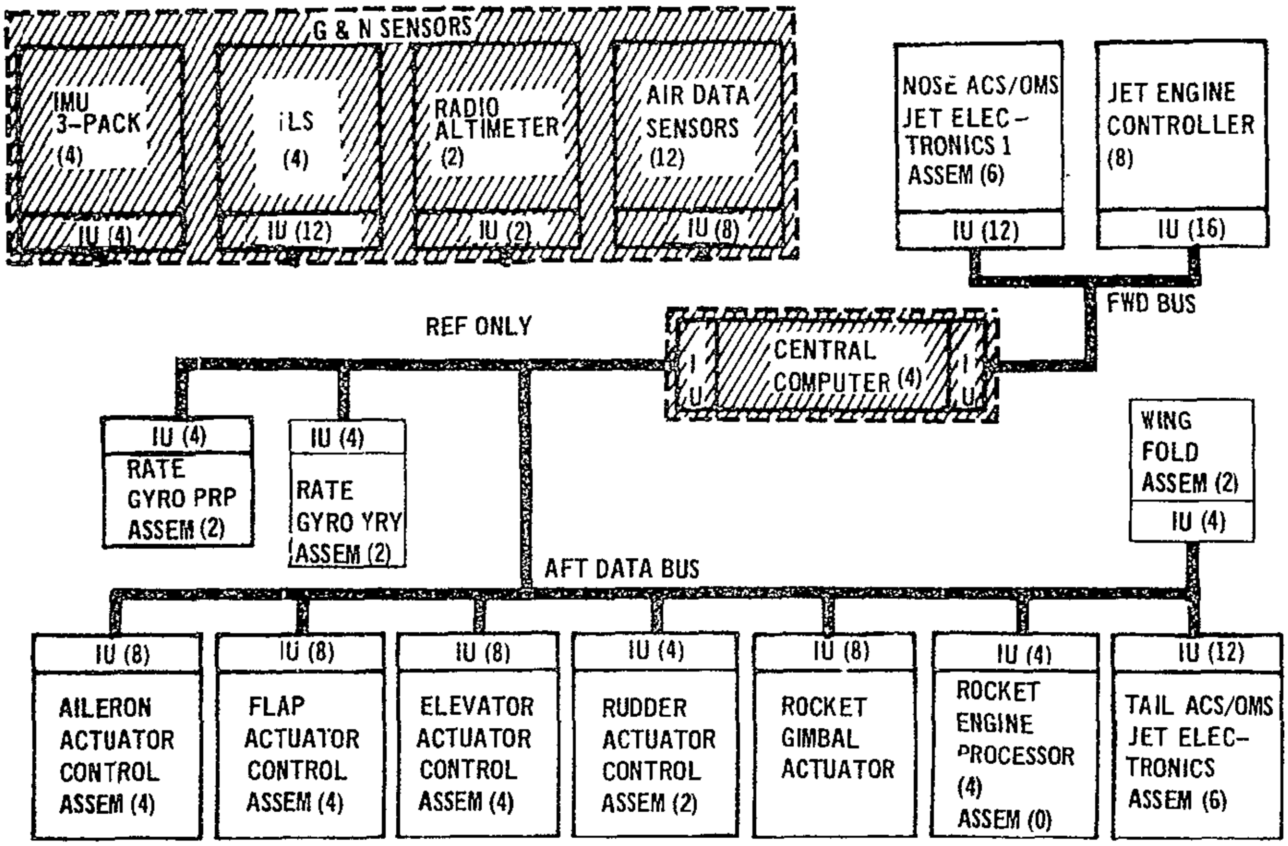
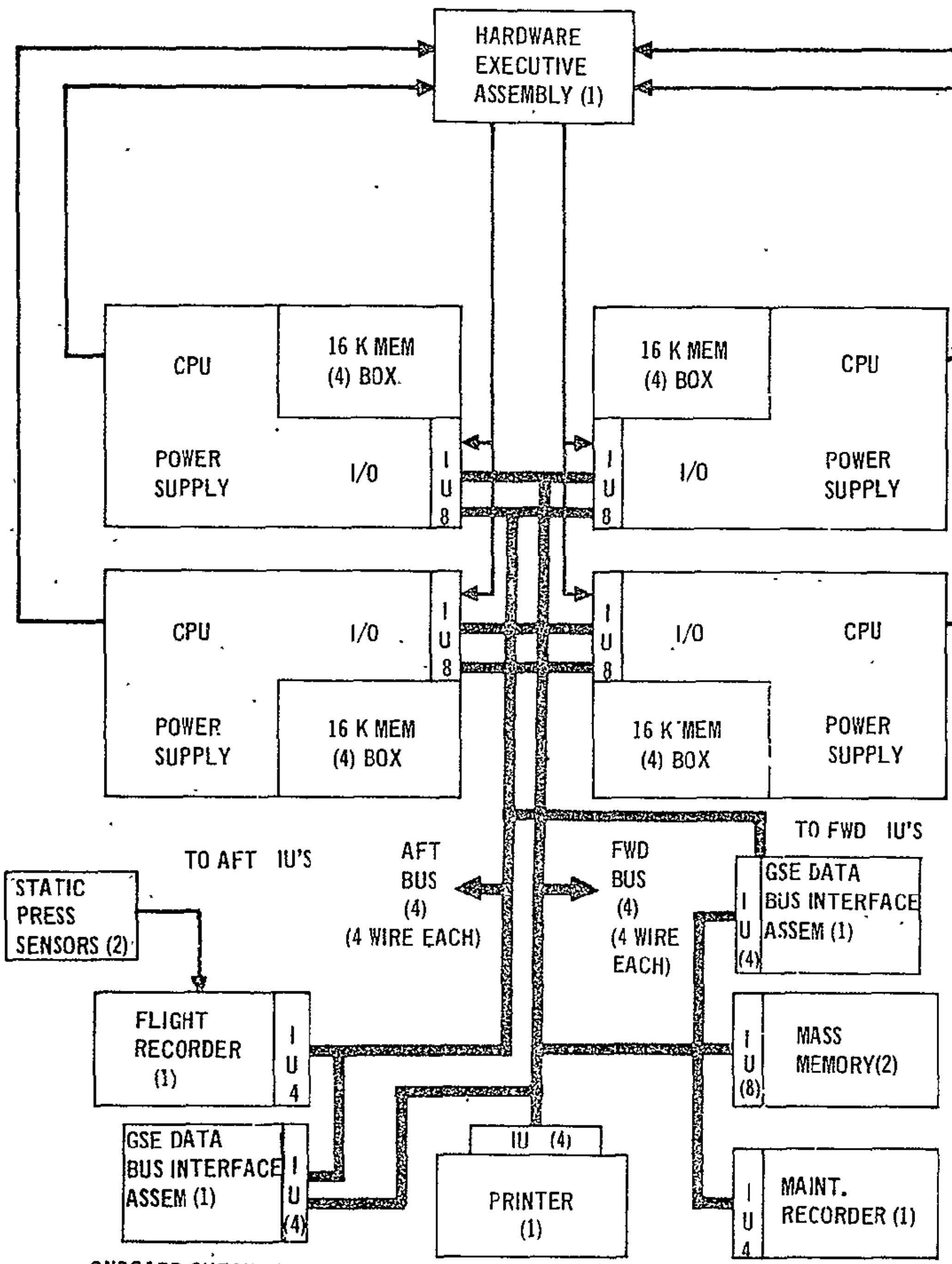


Figure 2-10 Orbiter Flight Control Electronics



- ONBOARD CHECKOUT, CONFIG. SEQ, FLIGHT CONT, G&N COMPUTATION ALL INTEGRATED
- FOUR UNIPROCESSORS, ONE IN CONTROL, AUTOSWITCH BY EXEC UPON FAILURE.
- STANDARD DIGITAL INTERFACES TO ALL LRU'S VIA DATA BUS

Figure 2-11 Orbiter Data Management Subsystem

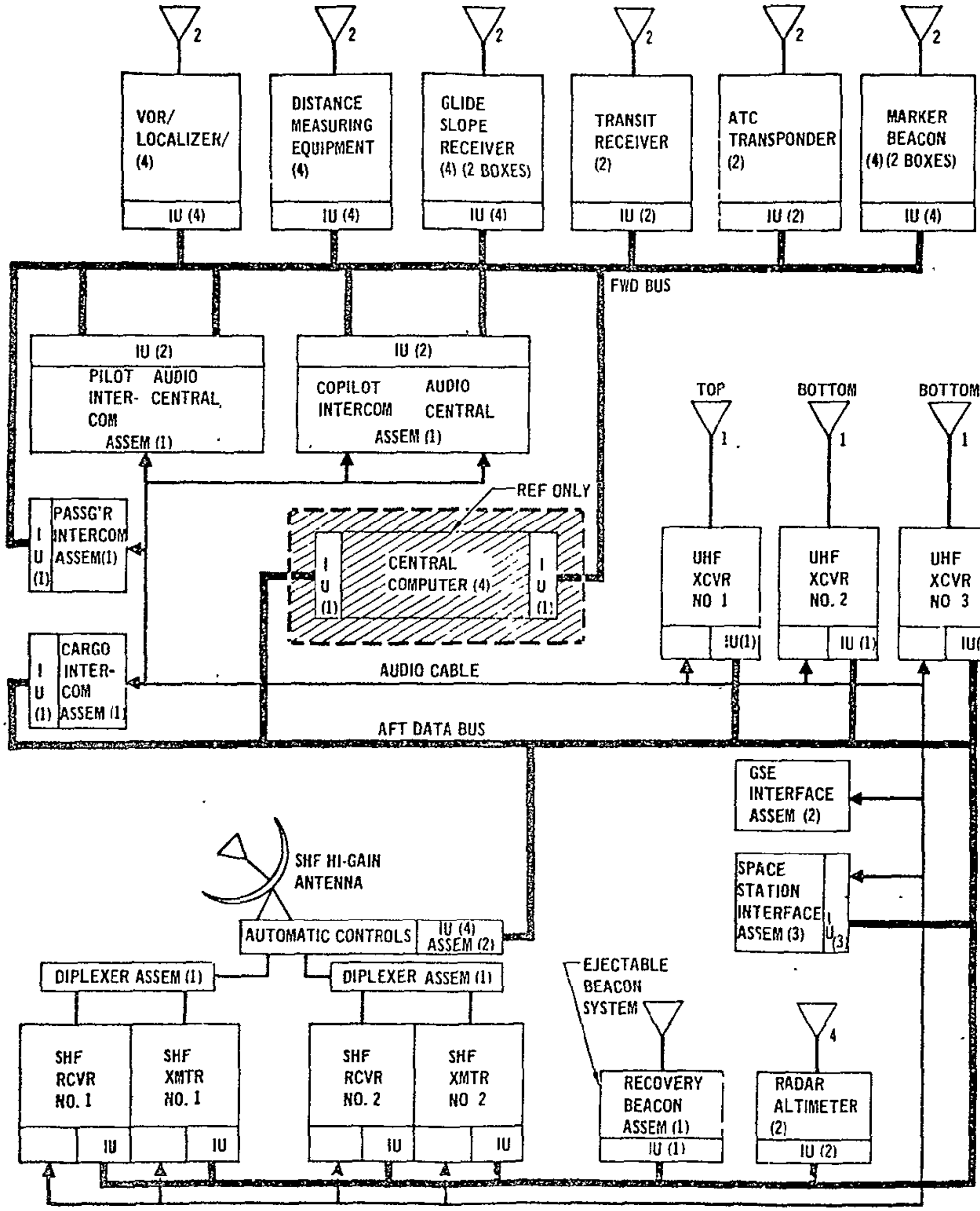


Figure 2-12 Orbiter Communication and Navaids

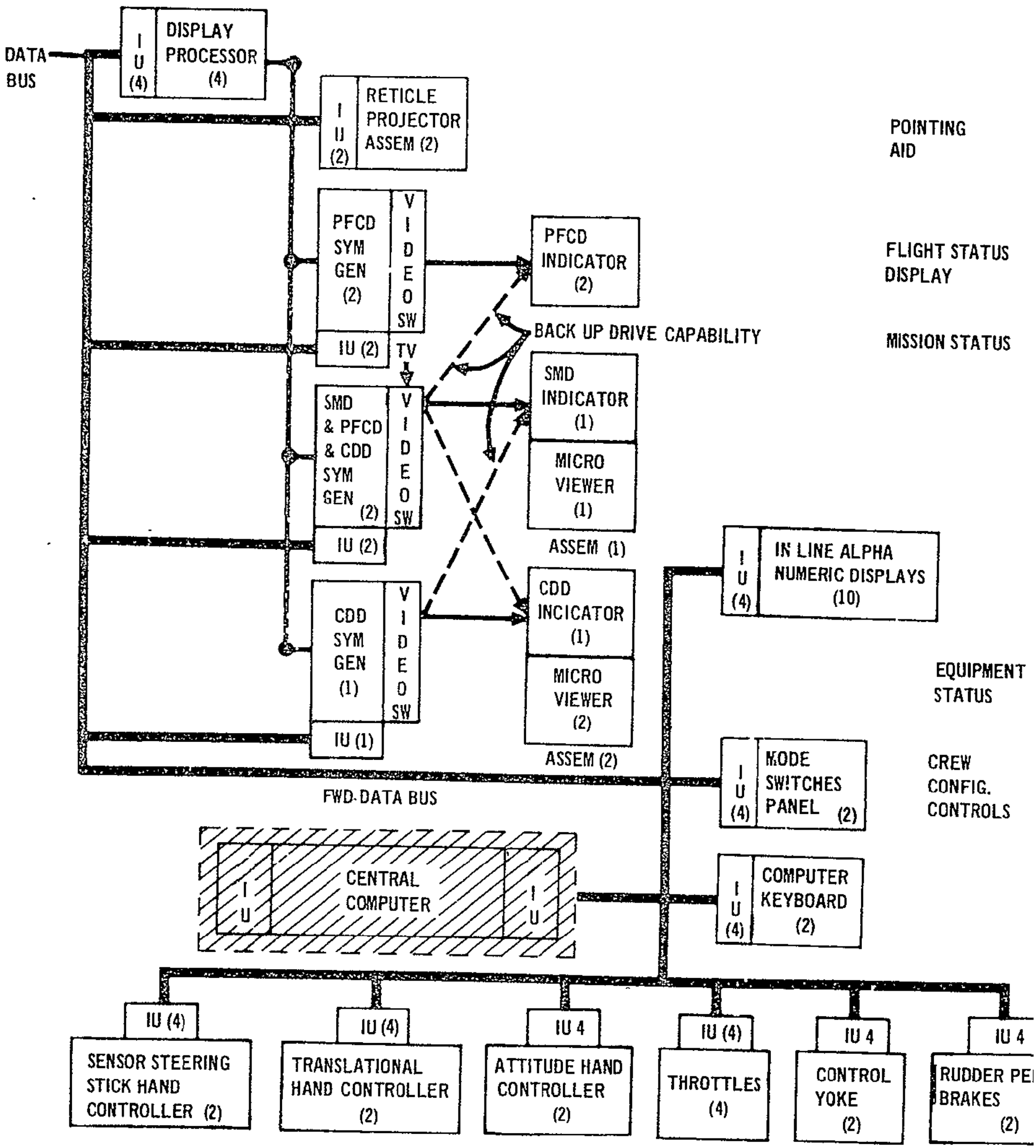


Figure 2-13 Orbiter Controls and Displays Subsystem

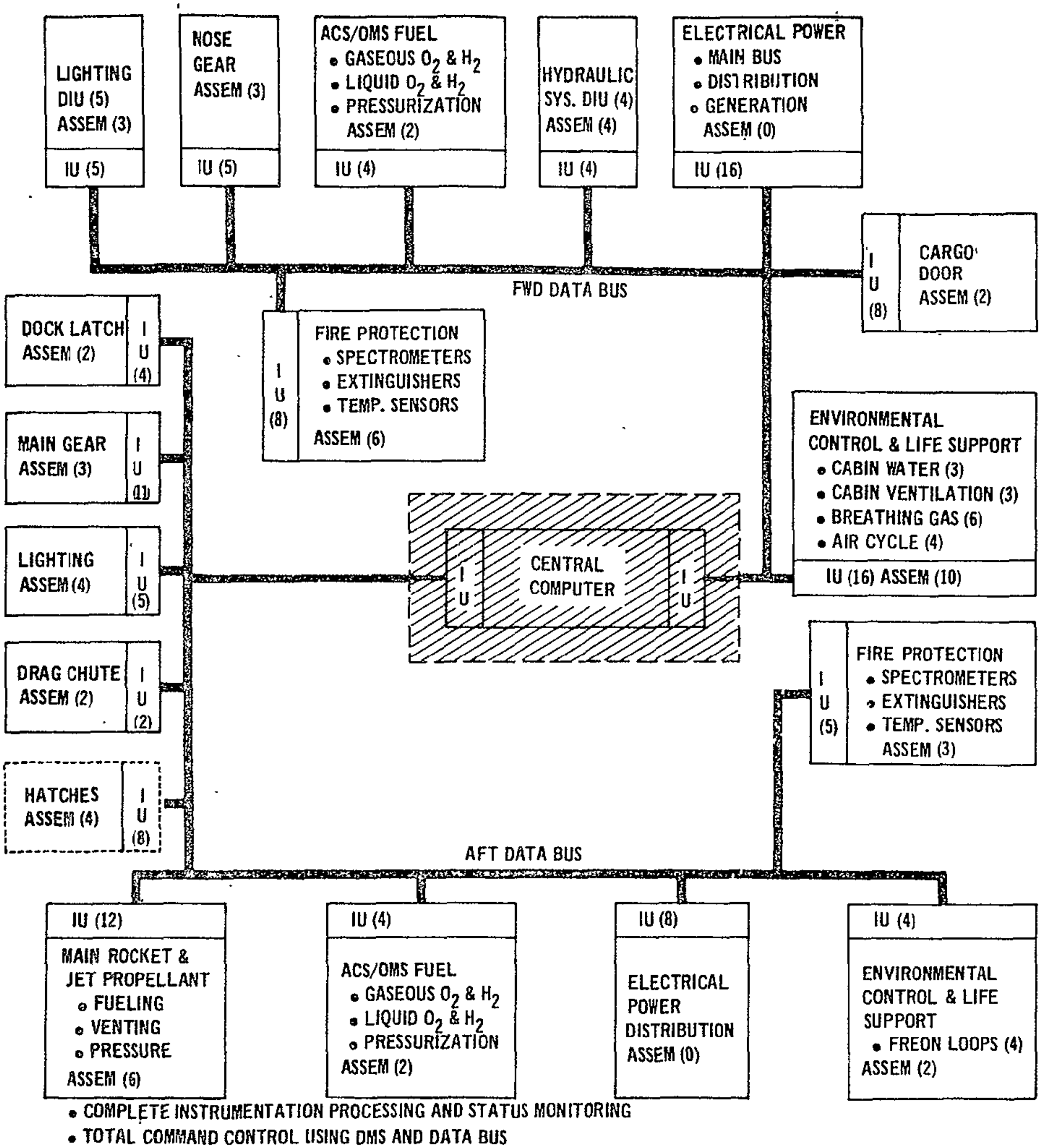


Figure 2-14 Orbiter Configuration and Sequence Control Subsystem

2.4 Computer Systems (Booster and Orbiter)

The functional and operational requirements of the Space Shuttle are controlled and monitored by the central computer subsystem. The central computer performs all computations with the exception of those which are delegated to the engine computers (main and jet), the display computer and the IMU computer. In addition, certain computations, primarily in the areas of self-test and bus transmission, are performed at the IU's. Figure 2-15 shows the central computers and the supporting functionally oriented computers. The diagram is not intended to depict the physical interconnection of the system, but rather to show the division of computational capabilities and the degree of decentralization which the baseline system exhibits. In addition, the diagram makes no attempt to show redundancy or data bus interconnection considerations.

The subsystems and the functional computers are all capable of communicating with the central processor but not with each other. This degree of decentralization has been adopted to accommodate the timing requirements as presently defined for the system.

2.4.1 Engine Computer (Jet and Main)

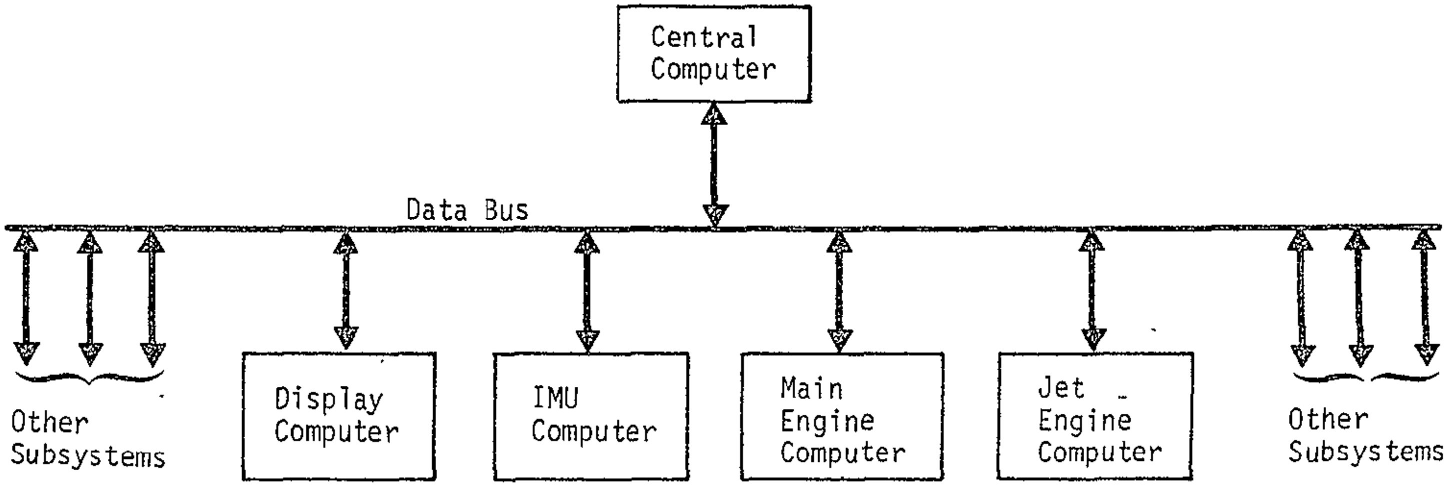
The central computer is responsible for the derivation, definition, and transmission of commands to the engines. The engine processor is responsible for direct control of the engine and of the processes associated with engine operation. Another responsibility of the engine processor is to monitor engine performance and to formulate engine status information.

2.4.2 Display Computer

This computer has the responsibility of generating the displays for the cathode ray tubes and microfilm-viewers which interface with the crew. The central computer prepares a list of parameters and transfers them to the display computer without any regard for how they are to be displayed. The display computer extracts the parameters needed for a particular display, orders the information within the display and transforms the information (created vectors, pictorial presentations, alphanumeric text, etc.) so that it is compatible with the electrical drive characteristics of the display hardware. Control of display modes and direction of information to appropriate displays is also the responsibility of the display computer.

2.4.3 Inertial Measurement Unit (IMU) Computer

In order to ascertain the characteristics of the IMU computer needed to satisfy the requirements imposed by the strapdown guidance sensor, it is assumed that the device is a "6 pack" and that the following computational functions are associated with the device:



Note: Diagram does not identify any of the IUs which have computational capabilities.

Figure 2-15 Distribution of Computational Capabilities

- Sensor Compensation Calculations

- Sensor Performance Monitoring - These computations involve the use of test signals provided by the sensors as well as end to end tests of sensor behavior normally utilized with redundant sensors.

- Transformation of good sensors to 3 axis output.

- Transformation from Body to Inertial Coordinates

- Computation of Inertial Velocities and Body Rotation Rates

- Computer Self-Test

These computations are performed every 20 msec. The required results of these computations are transferred to the central processor at the same rate. This transferred data includeds the following items:

- Inertial Velocities (3)

- Body Rotation Rates (3)

- Direction Cosines (6)

- Sensor Status Information

- Computer Self-Test Results

2.4.4 Central Computer

The central computer complex provides the capability of onboard computation, data processing, data storage, sequencing and control for the Space Shuttle. The central computer complex performs the computations and processing for guidance, navigation, flight control, flight management, communications, checkout, display and crew control functions. It provides a digital interface for the control of and communication with the subsystems via the data bus. It also provides the man-machine interface to permit the crew to select the computers.

In addition, the central computer complex provides the data processing, storage and program loading capability to facilitate prelaunch and onboard checkout. Data will be recorded to aid post flight checkout and refurbishment.

The central computer complex consists of four independent but identical general purpose computers. Each computer includes an input/output control unit (IOCU). Each IOCU interfaces with four data buses and can control one of the four data buses at a time.

The computer can have access to any subsystem or line replaceable unit (LRU) via the IOCU and one of the buses.

During non-critical phases of the mission, one or two computers are active and operating. The computer monitors subsystems or LRU's connected to any one of the data buses. During critical phases of the mission, three or four computers are active and under the control of the identical program. Each computer monitors all of the avionic system; but only the computer, which is designated as the operating computer, issues subsystem commands to the subsystems. The other three computers receive all the data and issue IU interrogations, but they do not issue commands to the subsystems.

Each computer consists of a central processing unit, an IOCU, and memory modules. The processing unit performs all the control, arithmetic, logic, and data manipulating operations within the computer. The control section of the CPU sequences the processing unit in accordance with the stored program instructions and the status of a variety of internal interrupts. The arithmetic units performs all arithmetic and logic functions.

The data transfer between the subsystems and the central computer complex is controlled by the Input/Output Control Unit (IOCU). The Central Processing Unit (CPU) initializes the IOCU function under CPU's program control. Once the IOCU is initialized, it performs all the data transfer autonomously. Upon completion of the input/output cycle, the IOCU notifies the CPU and prepares for the next data transfer cycle. The functions to be performed by an IOCU include:

- Provide timing and control of the data bus.
- Provide data buffering and word formatting capability.
- Perform addressing and interrogating functions of the subsystems.
- Provide error detection and correction for the data bus.
- Perform reasonableness check of the data format.
- Send commands from the central computer complex to the subsystem.
- Handle status information from subsystems to the computer.
- Provide the buffer control for all the data inputs and outputs.
- Multiplexing and de-multiplexing data.

- Distribute mission timing information to the subsystems
- Perform gather/scatter operation of data from/to main memory.

In order to maintain overall data management system control and to resolve any conflicts that may arise among the four computers, a System Control Unit periodically compares key data from all four computers. It provides the hardware capability necessary to monitor and control the central computer complex which, in turn, controls the data bus and the subsystem IU's. It provides the necessary status displays and manual control for the crews who provide the final control of the system. As described previously, the data management system consists of four identical and independent systems, each consisting of one CPU, and one IOCU which interfaces with the four data buses. These four redundant systems are under the control of the System Control Unit.

Normal operation of an IU requires the central computer to preprogram or condition the IU via the Data bus. The IU is then capable of self interogating various parameters within its subsystem, and checking them against defined limits. Out of limit conditions are sent to the central computers for further action. This is accomplished through the built in test equipment (BITE).

The central computer can command an IU to take a designated (or preprogrammed) action. This action can be the result of a

time interval
crew command
mission event
mission time or
subsystem status

The command can be in the nature of an applied stimulus or the turning off or on of various controls within the avionics subsystem. The IU will normally notify the central computer that the commanded action has taken place.

The central computer may also interogate parameters within the system to supply information desired by other subsystems or to satisfy crew requests for status information.

The keyboard provides the data entry and control capability for the central computers. The printer provides the data printout capability for the central computers. The maintenance recorder provides the storage for information which facilitate maintenance, checkout and refurbishment of the Space Shuttle.

2.5 Guidance and Navigation Subsystem

The Space Shuttle baseline guidance and navigation subsystem is

comprised of four strapdown three-pack inertial measurement units (IMU) each with its own IMU processor, three combination rendezvous and docking laser/star tracker systems, two Transit receivers (part of the Communications and Navaids Subsystem), two horizon sensor assemblies, and two trainable television cameras. All of this equipment except for the Transit receivers is mounted in two sets on Navigation Bases to maintain accurate relative alignment. Two reticles are provided by the Display and Controls subsystem for alignment redundancy in conjunction with the horizon sensors.

The star tracker will be used during the prelaunch phase to provide azimuth alignment of the IMU's through the transfer of a ground based optical reference. During the boost and entry phases of the Shuttle missions, the IMU's will be the sole source of navigation information. The IMU's will be mounted in pairs on each Navigation Base in a manner that will effectively result in six skewed gyro and accelerometer sensitive axes. This enables the utilization of six-pack redundant sensor processing techniques for each Navigation Base Assembly.

The star tracker, horizon sensors and Transit receivers will be used for attitude and navigation update information during the orbital phases of the mission. The horizon sensors will continually track the earth's horizon during all active orbital phases and supply angular information relative to the navigation base. The star tracker will track selected stars under central computer control during all orbital phases.

During rendezvous, docking and station keeping phases, the star tracker must be time shared so that relative navigation updates may be provided using the laser. The Transit receiver will provide navigation updates whenever a Transit satellite is in view of the Shuttle.

The laser rendezvous and docking system will provide relative navigation information (range and angle) for those missions when rendezvous with a cooperative target is to be accomplished. An interface will be provided to accept data from a rendezvous radar for uncooperative targets although this radar is not a baseline requirement.

For cruise navigation during ferry flights or between entry and landing, the IMU will be the primary source of information with navigation updates provided by the VOR/DME radio navigation aids included in the Communications and Navaid Subsystem. During the landing approach, other radio aids (marker beacons, glide slope, localizer and the radio altimeter) will provide additional update information for navigation and steering.

2.6 Flight Control Subsystem

The Flight Control Subsystem provides the software and computations as well as the interface and sensing hardware to control all phases

of vehicle flight. After the desired flight pattern has been decided by the data management mission planning function or by the crew; the Flight Control Subsystem effects motion in the prescribed plan by control of lift, drag and thrust.

Basic programs consist of pitch, roll, and yaw axis control logic, attitude thruster control and dead band programming, orbit maneuvering thrust to effect rendezvous and deorbit, main engine thrust and vector control to fly a computed path in space and fault isolation and switching. Additional programs include altitude hold, altitude rate hold, heading hold, flare, decrab, stability augmentation, load alleviation (based on bending mode sensor inputs), attitude hold, stability and/or control augmentation, controller "feel" augmentation, flight and attitude control for minimum heating on entry, and flight warning (stall, flame out, gust, altitude rate, etc.).

The interface function transmits and receives signals from and to the central computer from and to all thrusters and control surfaces. The sensing function includes back up altitude and bending mode data from rate gyros, air data (total temperature, dynamic pressure and static pressure), and equipment status sensing to provide inputs for checkout and fault isolation.

The flight control system is integrated into the data management system to the extent that all software and computation is done therein. This leaves only the sensors, interface units and actuator drivers to be implemented. These devices include:

- Dynamic, static and temperature sensors for air data measurements
- Conventional rate gyros selectively located in order to minimize bending mode effects
- Surface actuator electronics, driving quad servo actuators
- ACS/OMS jet electronics providing thruster firing control
- Jet engine interface electronics
- Rocket engine interface electronics including gimbal actuators
- Associated IU's for fault isolation, sensing, and interfacing

2.7 Communications/Navaids Subsystem

The UHF portion of the subsystem consists of 3 solid-state transceivers and antennas, and operates in the 225-400 MHz portion of the spectrum. One antenna is mounted on the top of the fuselage

to permit coverage when approaching the space station in that direction as well as EVA in the upper hemisphere. Two antennas are flush-mounted on the bottom on the shuttle.

Each UHF transceiver contains a modem whose function is to provide the correct modulation and demodulation format for compatibility with MSFN, EVA, or military airports. MSFN and EVA use 100% modulation double-sideband AM (clipped-speech) while airports use conventional double-sideband.

In addition, the modem provides the capability of handling digital data using phase modulation techniques. Selection control of the modem format and interface with digital data source or destination is accomplished through the Interface Unit (IU).

The SHF (not used on the booster) portion of the system consists of 2 transmitters and receivers and a six-foot parabolic dish. In order to avoid RF switching with attendant single-point failure modes, dual feeds are used. The mechanical and electrical portions of the antenna pointing system incorporate dual gears, motors and electronics to provide inherent redundancy in a single LRU. A modem for each receiver-transmitter pair fulfills essentially the same function as the modem for the UHF transceiver except that in this case voice transmission will be accomplished using FM. It is anticipated that voice and digital data would be frequency-multiplexed on a common carrier. This decision depends a great deal on the amount of data to be transmitted to and from the Shuttle and upon operational requirements.

The satellite relay link is the primary communication and data link between the Space Shuttle and ground. A high gain communications antenna will be required which operates within the frequency bands of the civilian COMSAT program. Polarization is to be circular to match the Intelsat IV satellite antennas.

The antenna is stowed within the body structure and mechanically deployed into an operating position at the proper time. It is retracted and stowed prior to the Space Shuttle entering the atmosphere. The 6 foot antenna will probably be pointed by a computer programmed control instead of auto-track, although this subject will receive further attention.

The transmitters, receivers, and intercom boxes are tied together by a data bus and an audio cable. To interface with these buses, each LRU must be equipped with Input/Output coupling units. Analog information (voice), frequency-stacked on the audio cable, is selectively routed to the desired transmitter, receiver or intercom box. This control is accomplished through the IU via the Data Bus. Digital information is routed to or from each transmitter or receiver through the IU via the Data Bus. In addition, the self-test functions and diagnostic data for each LRU are handled by the

IU/Data Bus arrangement.

The Navaids consist of VOR/Localizer receiver, distance measuring equipment, glide slope receiver, Transit receiver (not used on booster), ATC transponder, marker beacon receiver. These equipments (with the exception of the Transit receiver) provide the capability for interfacing with the Air Traffic Control System during the landing and ferry phases. The Transit receiver provides the capability for updating the G & N Subsystem of the orbiter. These equipment interface with their antennas and with the Data Bus.

2.8 Non-Avionics Subsystems

The Non-Avionic Subsystems provide the software and hardware to affect sequencing and control of non-avionic subsystems either automatically or under manual control. Its functions include (1) determination of what the non-avionic vehicle configuration should be as a function of time, (2) what the configuration actually is, (3) computation to minimize differences and (4) actuation means for effecting control. Checkout, initialization, all flight regimes, standby (in orbit), go around, post flight, safing, and ferry mode sequences for

Landing Aids

Fueling and Pressurization

Hydraulic System

Electrical Power Generation

ECLS

Payload/Space Station/Ground Interfaces

Lighting

Hatches and Ground Access Doors

Fire Protection

The Non-Avionic Subsystems use computer commands decoded by distributed IU's which are then converted to appropriate power distribution substation circuit breaker on/off signals. This control can be a response to a programmed sequence, a crew decision, or due to a failure detection. In the latter case, the subsystem detects the fault and affects the switching of power to a redundant non-avionic component.

No more than one IU or power bus will connect to a single active element. In the vehicle subsystems, therefore, only one IU appears

for each redundant element (valve, pump etc).

If a local IU should fail, such an element would be taken off line by the computer using the appropriate power control station.

Software, computations, and fault detection in general are accomplished by the central computer. Actual implementation of non-avionic functions thus reduces to (1) configuration sensing (what is the present status?), (2) interfacing between the data list and the non-avionic subsystem including command decoding and (3) actuator driving.

2.9 Displays and Controls

The displays and controls for the Space Shuttle vehicle utilize state-of-the-art equipment and techniques to provide the autonomous mission operation capability by two crewmen, with emergency operation by a single crewman, without a crew task overload. Electronic multi-mode displays allow the presentation of the data of all the different flight regimes in a limited cockpit area and pilot view cone. The crew task load is reduced by functionally grouping system management panels and designing to manual mode selection with automated sub-mode sequencing. The crew has manual override of all automation. Conventional dedicated displays are also provided for certain failure mode analyses and aid in meeting the maximum instantaneous display requirement.

The basic mission operational data presented to the crew includes:

- vehicle attitude reference
- horizontal or vertical situation
- operational data from onboard systems
- crew/computer communications
- status monitor of on-board systems

The controls and displays presented to the crew by direct view include the inline alphanumeric displays, dedicated displays, two reticle projectors, and four cathode ray tubes.

The controls are basically catergorized as attitude and velocity control, computer access, and subsystems, mode, or sequence selection and control. The functional arrangement of the cockpit displays and controls reduces crew training and eases crew operations.

3.0 SPACE SHUTTLE TEST RELATED ACTIVITY FLOW

Figure 3-1 is a flow diagram of the operational Space Shuttle mission, showing the activities which are significant with respect to test and monitoring. Figure 3.2 is a tentative schedule of these activities. In addition to the activities shown, there will be prior development and acceptance test activities at prime and subcontractor facilities. The development and acceptance activities cannot, at this time, be defined with any significant degree of confidence other than as dictated by the operational requirements. It is believed that the pre-operational test activities should be accomplished using (and developing) the test philosophies (and test language) provided for the operational system. For the purpose of defining test language requirements if should therefore be adequate to emphasize the operational requirements.

A Space Shuttle mission cycle may be assumed to begin with the separate vehicles and modules undergoing independent preparation and testing.

3.1 Orbiter Checkout

3.1.1 Orbiter Subsystems Checkout

The Orbiter, located in a VAB low bay area, undergoes extensive testing of its subsystems after any previously identified defective components or line-replacable-units (LRU's) have been replaced from spares. During these tests, most subsystem interfaces will be provided by onboard IU's, BITE, and interacting subsystems. External interfaces (radiated, pneumatic, booster, payload, etc.) of a non-hazardous nature will be provided by GSE and simulators. Many of the subsystem tests may be performed concurrently with other subsystem tests when such simultaneous tests are non-interfering. In addition, there should be flexibility to reschedule tests of subsystems in order to provide time efficient progression in spite of modifications, failures, repairs, etc. This rescheduling might be accomplished semi-automatically by informing the processing system of anticipated availability of subsystems and LRU's. The processing system can then continually predict schedule problems, indicate critical schedule paths, and identify alternative sequences.

3.1.2 Orbiter System Checkout

The Orbiter system level tests should be accomplished after all subsystems have successfully passed their checkout routines. The major purpose of this test phase is to assure that functionally redundant and interacting subsystems perform compatibly and that the central computer systems properly assess and respond to simulated subsystem inputs including anomalies and malfunctions, and operator (pilot) inputs. A portion of this test phase will be a simulation of the flight mission phases where the orbiter is independent of other mission modules (i.e. Booster separation thru landing).

3.2 Booster Checkout

3.2.1 Booster Subsystems Checkout

The Booster, located in a separate VAB low bay area, will undergo subsystems testing in the same manner as the orbiter.

3.2.2 Booster Systems Checkout

The Booster system level tests will be similar to orbiter systems tests, with the following differences. It will not be necessary to simulate Payload system interfaces. Launch GSE interfaces will be somewhat more extensive than for the orbiter. Mission simulations will be less extensive than for the orbiter due to the absence of orbital phases.

3.3 Payload Systems Checkout

Complete checkout of the payload will be accomplished in a separate facility, and will be somewhat unique for each mission depending on the specific payload definition. It is not necessarily constrained to the same turn-around cycle as the orbiter and booster, and it will not derive as many benefits from repetative mission performance assessments. Its testing will undoubtedly be more extensive in the area of environmental compatibility. For manned (passenger) payloads; life support, egress, passenger communications, and passenger interactions are considerations in the Payload Systems Checkout.

3.4 Mobile Launcher Checkout

The Mobile Launcher Systems will be refurbished then checked out in the VAB high bay to determine their readiness to accept the booster and orbiter. This testing will include the verification of electrical power, unbilical, egress, propellant, gas, RF, communications, environmental control and life support and other interfacing systems. The Mobile Launcher is not necessarily constrained to the same turn-around cycle time as the Booster and Orbiters. It might have significant launch to launch variations in the areas of payload related support functions.

3.5 Shuttle Systems Tests

Combined systems tests will be performed in the VAB high bay area where the Booster and Orbiter are installed and mated on the Mobile Launcher. Primary emphasis of the mated tests in this area will be the confirmation of interfaces and interactions of the Booster, Orbiter, and Mobile Launcher, and will include testing of umbilical systems, propellant and gas control and monitoring signal interfaces, egress systems, separation systems (launch and in-flight), communications systems, data buses, etc.

FOLDOUT FRAME 1

FOLDOUT FRAME 2

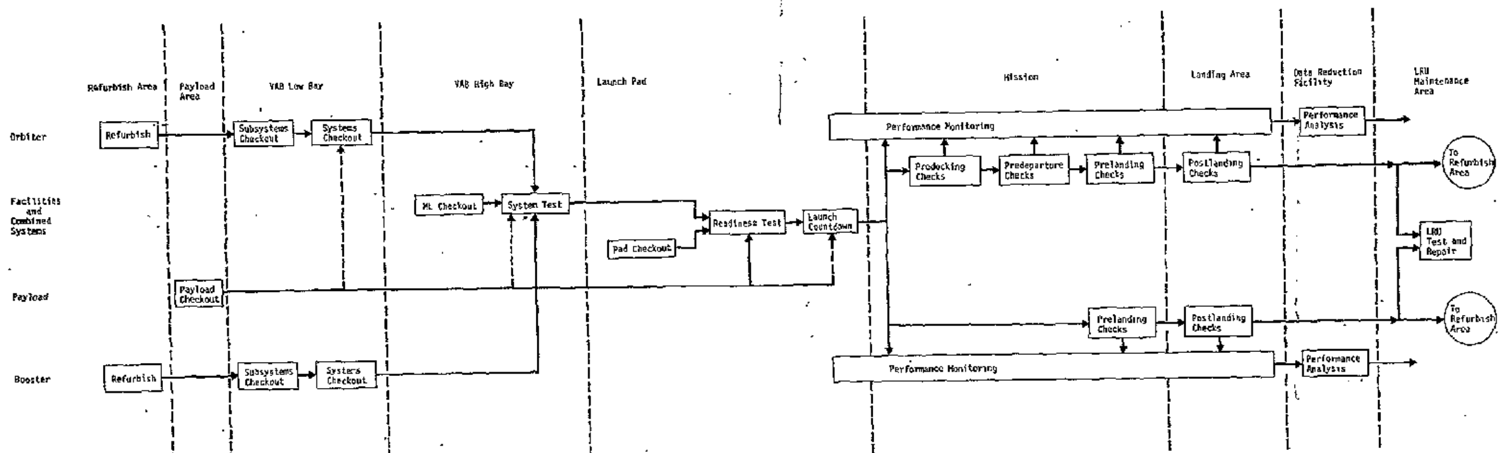


Figure 3-1 Shuttle Test Activity Flow

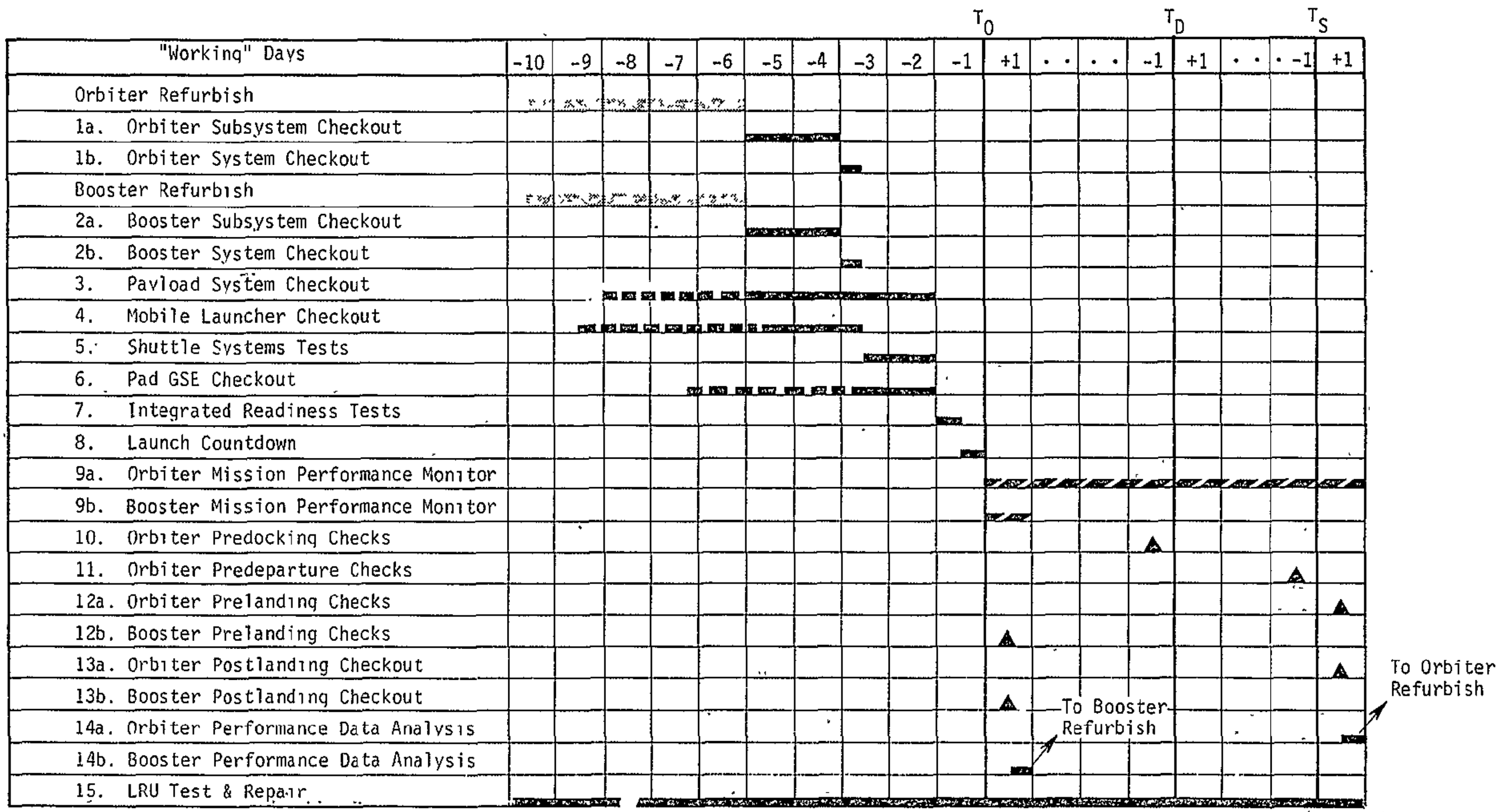


Figure 3-2 Shuttle Test Activities Schedule

3.6 Pad GSE Checkout

Tests to verify readiness of the Pad GSE will be performed prior to arrival at the pad of the Shuttle Orbiter/Booster configuration mounted on the Mobile Launcher. This checkout will include verification of signal interface and transmission capabilities, propellant and gas control and safing systems, and numerous related hazard and fire sensing systems.

3.7 Integrated Readiness Tests

After the Mobile Launcher with the mated Booster and Orbiter have been checked out then transported to the launch pad, progressive tests will be performed as the launch pad interfaces are verified and connected and as the payload is installed. Simulated countdown segments may be performed and acquired performance data analyzed. The redundant systems of the Booster and Orbiter will be verified primarily by comparing the performance of the redundant systems and verifying status of subsystem BITE. Some simulations may be run with pilot and crew functions being performed from the remote control center.

3.8 Launch Countdown

After Launch Readiness has been established, and all data from simulated sequences analyzed, the system will be placed in launch countdown mode consisting of time-constrained events (predominantly associated with propellant, gas, crew ingress, electrical power, environmental and mechanical systems) leading to launch within the launch window. During this period, performance of onboard systems and GSE will be monitored and compared with predetermined (dynamic) criteria. The countdown progression will include the initialization, updates and monitoring of guidance parameters in the redundant systems.

3.9 Mission Performance Monitoring

During all phases of mission performance, BITE status and the redundant onboard systems will be monitored and their performances compared. Any significant deviations of a single channel from other channels will result in the alteration of primary and alternate channel assignments, and system configuration for mission performance. All such deviations or anomalies and the resulting configuration decisions will be displayed and recorded. During these phases of the mission, it is to be expected that some capability will be provided for the pilot/copilot crews to request the display of performance data; however, it is not anticipated that the crew will generate or initiate "active" tests. The essentially passive performance and status monitoring phases for the Orbiter will include ascent, separation, orbit insertion, transfer, rendezvous maneuvers, post docking, descent, deorbit, reentry, and subsonic flight. The corresponding phases for the Booster will include

ascent, separation, reentry, and subsonic flight.

3.10 Pre-docking Checks

Prior to docking, the Orbiter may perform specific test sequences to determine and/or verify sensor and control paramenters in preparation for the final docking maneuvers. These tests and calibrations would be pre-programmed and might be either automatically or manually initiated.

3.11 On Station Checks

Pre-departure checks will be performed after the Orbiter has been docked to the Space Station for a period of time and before beginning the return mission. These checks will be performed to verify the operability of subsystems which have been deactivated or static for extended periods of time. It may also be desirable to checkout actuators, control surface movement, nav-aids, turbine engine controls, etc. that will be required for the first time during the return mission.

3.12 Pre-landing Checks

These similar checks for the Booster and for the Orbiter will include predominantly status checks by the crew, comparison of redundant G & N, Nav-aid, and visual course data, and confirmation of vehicle approach and landing events and configurations as they occur.

3.13 Post-Landing Checkout

Some checkout sequences will be performed before systems are shut down for scheduled refurbishment and maintenance. The primary purpose of this checkout will be to give greater assurance that all malfunctioning LRU's can be identified and replaced during the refurbish and maintenance period prior to the beginning of subsystem tests in the VAB Low Bay area.

3.14 Post-Mission Performance Data Analysis

Performance assessments will be based on the computer analysis of recordings made during all phases of the vehicle's mission from lift-off through post landing checkout. This analysis will identify malfunctions and trends, and in some cases specific LRU's to be replaced. In other cases, required fault-isolation tests may be identified to be run for the purpose of isolating defective or suspect LRU's. This data analysis task should be very valuable in identifying required maintenance actions and scheduling of the refurbishment and retest activities for turn-around. The analysis should be completed as soon as possible after the Post-landing checkout.

In addition to the indication of malfunctions and anomalies, the performance of system elements in unique environments can be determined for use in future flight programs (e.g. accelerometer bias at 0 "g", thrust values in vacuum).

3.15 LRU Test and Repair

Line Replacable Unit maintenance testing will be performed on all spares and all items removed from flight vehicles for any reason. Removals may be for suspected or verified failure, modification, scheduled maintenance or recertification. After thorough testing (with repair, recalibration and retesting if required) the recertified LRU's will be placed in carefully controlled spares storage. They may be retested after removal from spares for reinstallation in flight vehicles. Some items (such as INS sensors) may require conditioned or powered storage, continuous monitoring, and/or cyclic retest. The LRU's will include all avionics sensors and "black box" subsystems whether with or without IUs and the IUs themselves. It has been estimated that the fleet of Orbiters and Boosters will contain over 8000 LRUs. Due to the stringent environments encountered by the Shuttle Vehicles and the high integrity requirements for all vehicle systems, it anticipated that removal rates of LRUs will be much higher than for military and commercial airlines avionics units. The resulting test activity for LRU maintenance will undoubtedly be a very large operation, perhaps larger than for the largest airlines today.

Preparation and maintenance of test programs for the many LRU types will be a correspondingly large task. Figure 3-3 is the anticipated LRU maintenance flow diagram, and shows several alternatives which would greatly influence the magnitude of the activity.

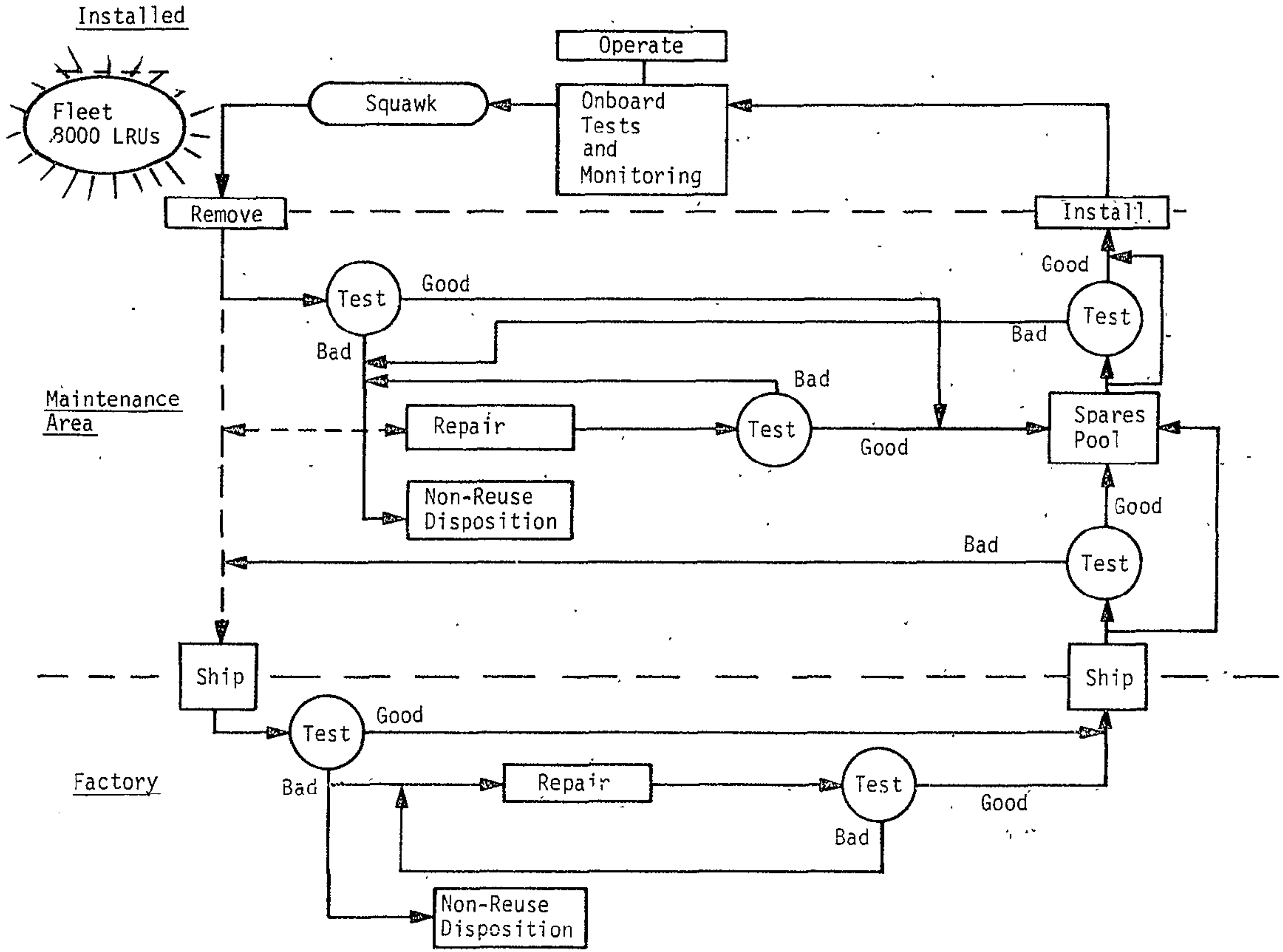


Figure 3-3 LRU Maintenance Flow

4.0 PROJECTED TEST SYSTEM CONFIGURATIONS

The ground rule; "maximum onboard autonomy" places the task of vehicle checkout and verification on the onboard central computers. Trade offs must be made however on the practicability of onboard control of vehicle test versus the necessity of ground support equipment (GSE).

The nature of control and monitoring of the GSE can (will) overpower the onboard central computers. Thus this report assumes GSE with its own control and monitoring equipment. Commands for GSE action are received from the vehicle via the data bus and Status information is returned to the vehicle.

The programming of the GSE will utilize the same test oriented computer language as that used by the flight engineers in programming the vehicle.

4.1 Checkout, Verification and Testing

Checkout, verification and testing occurs in all phases of Space Shuttle operation. These phases are:

- Maintenance
- Prelaunch
- Launch
- Flight
  - Ascent
  - Orbital*
  - Rendezvous*
  - De-orbit*
  - Return
  - Entry
- Landing
- Post landing

*Orbiter Only

Checkout takes the form of:

- Performance Monitoring
- Command and Response
- Verification

Performance monitoring is the normal mode of checkout within the Space Shuttle. This consists of reading vehicle parameters or combinations of parameters and evaluating them against predetermined standards. This is largely accomplished by BITE.

Abnormal performance during flight is indicated (where applicable) on cockpit panel lights and/or displayed on CRTs. The central computer performs fault analysis and removes malfunctioning subsystems from service and indicates this fact on the maintenance recorder.

Command and Response testing is the application of a stimulus and checking the response. Since it is not desireable to interupt active flight systems by applying calibration or stimulus signals, this method of checkout will be used primarily during maintenance, prelaunch and postlanding phases.

Verification provides the Space Shuttle operating and test personnel with a means of checking or verifying any parameter. Selection of parameters to be verified is accomplished by requesting status information via the onboard keyboard or the ground based data system.

4.2 Ground Support Equipment

A desired goal for the Space Shuttle is complete autonomy. However, as previously stated, it is believed that a large amount of ground support equipment (GSE) will still be required. This equipment will primarily be used during the maintenance and prelaunch phases.

Typical of the GSE necessary to support a mission are:

Vehicle monitor and back-up control

Computer language compiler, loader, and verifier

Data analysis

Vehicle ground power control and distribution

Guidance support

RF systems (including communications and navigation checkout)

APU service, control and monitor

Fuel Cell service, control and monitor

Hydraulic service, control and monitor

ECLS systems control and monitor

Egress systems, control and monitor

Payload, control and monitor

Air Data, supply and monitor

Propellant loading, control and monitor

Vehicle pressurization, control and monitor

Air conditioning, control and monitor

LRU maintenance

Rather than control and monitor these functions from the cockpits of the Booster and Orbiter it is believed desireable to have a ground control center. This control center could take the form of cockpit mock-ups with the CRT and lamp displays. Ground based computers would monitor the data buses and keep the ground (test) personnel informed of all onboard activities. The cockpit mock-ups will duplicate all onboard information. The pilot and/or copilot of either the Booster of the Orbiter could request GSE activities via the data bus. Testing of onboard systems during the pre-launch phase could be accomplished via the control center's computers connected to the data buses. This would eliminate the need for the ground checkout programs to be stored in the onboard central computers.

The data bus interconnection will also facilitate the transfer of compiled Test and Flight Engineer originated programs to the onboard central computers.

4.3 System Configuration

Based on our assumption that GSE will be used for Space Shuttle we must look at the non-flight phase of operation to envision activities at the base.

From the 14 day ground turn-around goal a recycle time-table Figure 3-2 has been derived. Automation must be employed in the GSE in order to accomplish and monitor all tasks. The GSE must be flexable and able to mate with the vehicles in several areas.

The preliminary GSE would look like that shown in Figure 4-1. Note that a control center is provided for both the Booster and Orbiter. As the vehicles are moved from the landing area to the launch site the operators at the control center are appraised of vehicle status and can assist or take any necessary actions.

At the landing site, information from the onboard maintenance and flight recorders can be transferred to the control center for analysis. From this information plus a crew debriefing the refurbishment task can be organized.

During maintenance all defective or suspected LRUs are replaced. A thorough checkout of each vehicle is then made in the VAB area to verify the operational status using stimuli and control commands from the ground control center. The Booster and Orbiter are then mated and system readiness checks are made.

The vehicle is then moved to the launch-pad where the ground based control center monitors the pre-launch preparations and controls fueling and other start-up functions. Such functions are shown in Figure 4-2.

Launch, while controlled from the vehicle, is monitored by the ground control centers. Here concurrance is noted with the count-down sequence. As onboard problems are indicated on the ground monitors, recommendations or solutions can be relayed to the crew for action via the communication links. The prime control for launch will originate in the vehicle cockpits with the pilot of the Booster in command for launch. After launch the ground control centers can be reassigned to other vehicles, compiling tasks, and data analysis.

4.4 LRU Maintenance

An extensive LRU test and repair activity is anticipated for the Space Shuttle and the supported Space Stations and/or Bases. While this activity could conceivably be supported by the same control center computers used for the vehicle checkouts, it is not believed that such a configuration would be practical. The variability of testing activities and the very large number of test programs for LRU's would create many additional programming problems for the control center processors and peripherals. A separate, multi-station test system with its own computer(s) and peripheral system(s) is assumed. It is very probable that this function could be accomplished using existing computer systems such as CDC 160G's XDS 930's, DDP 224's, etc. Such a test system would be very general purpose and would have the capability to generate and measure a far wider range of parameters than the onboard IUs. The test language requirements for this system would include extensive signal characteristics specification capability.

4.5 Payload Support Equipment

More complex payloads will include experiment systems, passenger environmental control and life support systems, etc. Since these systems will sometimes require unique and extensive testing facilities, it is anticipated that a separate area with payload peculiar checkout systems will be provided.

As with the LRU maintenance area, some existing test systems and computers may be used to good advantage. The test language requirements for this area should be similar to those for Space Shuttle systems testing, but perhaps requiring more detailed

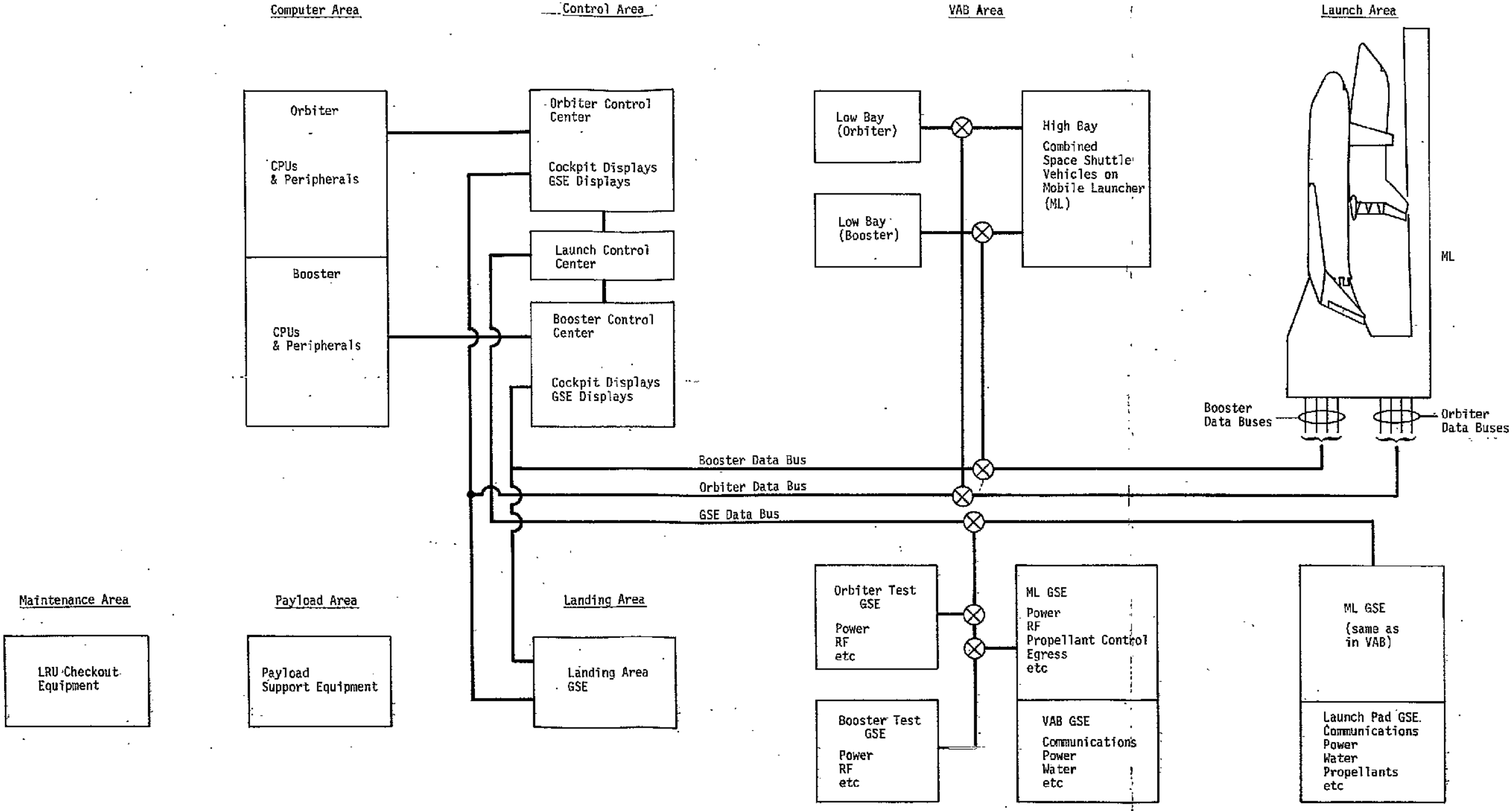


Figure 4-1 Base Support System

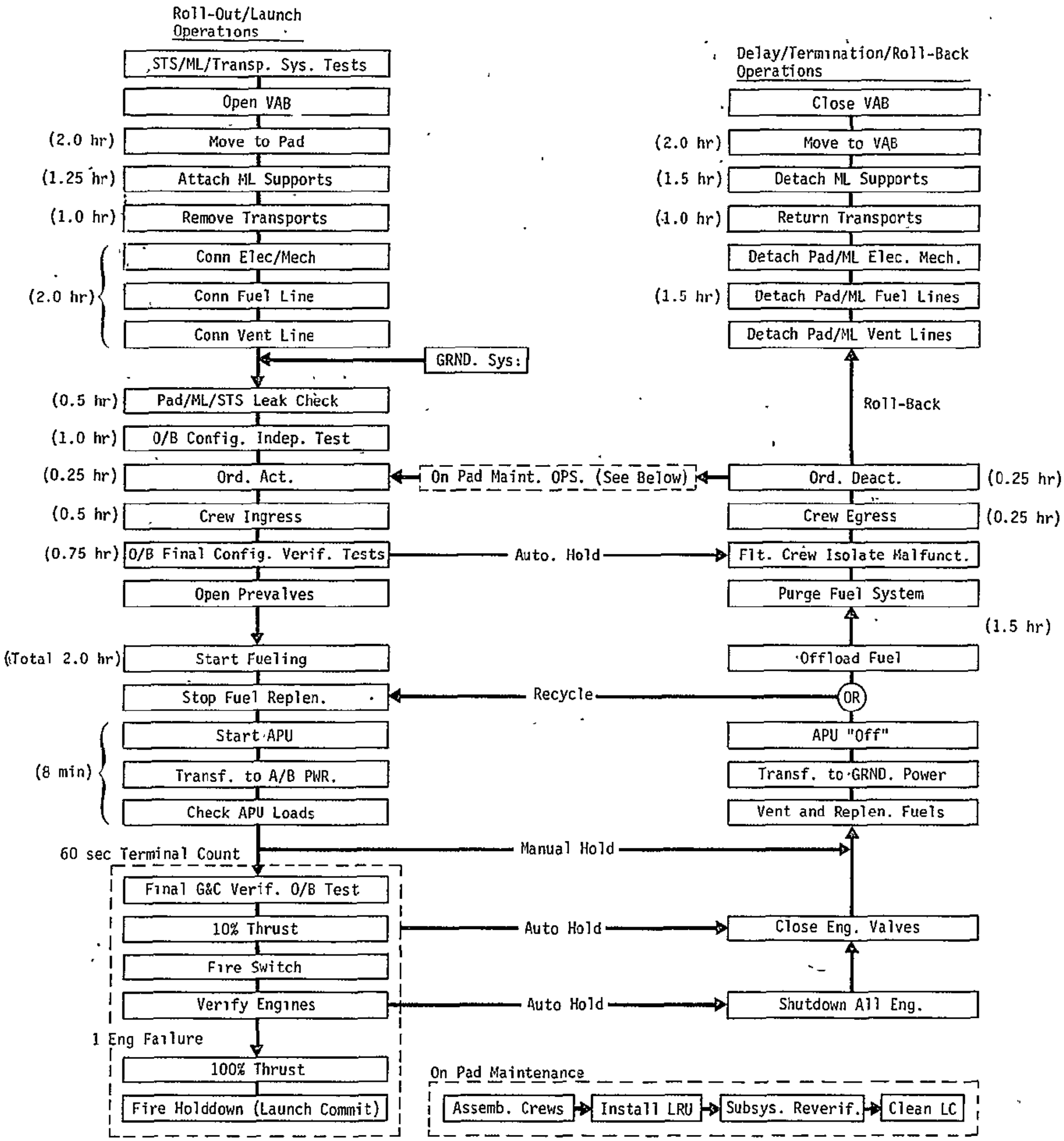


Figure 4-2 Roll-Out/Roll-Back Sequence

capabilities for specifying signal characterics.

4.6 Compiling/Translating Systems

With the system configurations suggested for the support of landing to launch operations, either the onboard computers or ground control center computers can control and monitor parameters associated with the vehicle or the GSE. A common language for these test and control activities is therefore essential. The ground computers should have the capacity to compile (or translate) all programs, and to load translated programs to the onboard computers as required. Comprehensive librarys of programs, subroutines, etc. for all vehicle configurations would be maintained in the ground computers and would be available during the translation phases. Interactive test preparation consoles might also be provided in this area.

The compiling/translating functions for the LRU and Payload areas might best be provided by their independent computer system. A major consideration in this decision will be the extent of test system and test language extension required to support these activities, and the greater probability of test system, test language, and test program changes which would impact the compiler/translator and associated library systems.

5.0 USER DEFINITION AND ASSOCIATED LANGUAGE CONSIDERATIONS

A test and flight engineer oriented language implies a language specifically fitted to the education, technical vocabulary, experience and training of test and flight engineers. Defining their characteristics and needs is therefore a very important aspect of the language study. Since there are others who will also use the language (test and flight equipment designers and programmers), it is also necessary to consider their capabilities and vocabulary. Training of any of these users is acceptable but should be minimized. It is the intent of this section to identify and discuss the users and derive general conclusions which should form criteria for the new language.

Figure 5-1 is a flow diagram showing the typical flow of activities associated with the generation, maintenance and use of test procedures or programs for automatic test and control activities. This sequence of events and the personnel (users) involved might well vary depending on organizational structures, charters, safety implications, contractor interrelationships, etc. One conclusion of this study is that organizational structures and charters might well be influenced by the programming language itself. It is also true that the language can be designed to assist and perpetuate any specific organizational responsibility.

The flow of activities are similar and involve the same personnel when changes and modifications to existing test programs are processed.

Each "user" identified in the flow diagram is discussed in the following paragraphs.

5.1 The Test Writer

Based on design characteristics and performance requirements of the prime equipment, and with some knowledge of the test system capabilities, the writer prepares a step-by-step sequence of events to program the test system. He is forced to prepare this sequence in a language and format acceptable by the test system, therefore he must be an "expert" in usage of this language.

The program writer should also have a thorough knowledge of the prime equipment design, control, test and operation. The latter requirements make him a system applications specialist, requiring engineering training and test experience. Idealistically, he should have been involved in the system design. To the extent that he has to learn internal details of the operation and peculiarities of the test system and unfamiliar programming language features, his time is diverted from prime system study.

One objective then, of the test language should be to make it as easy to learn as practical for a test writer whose background is

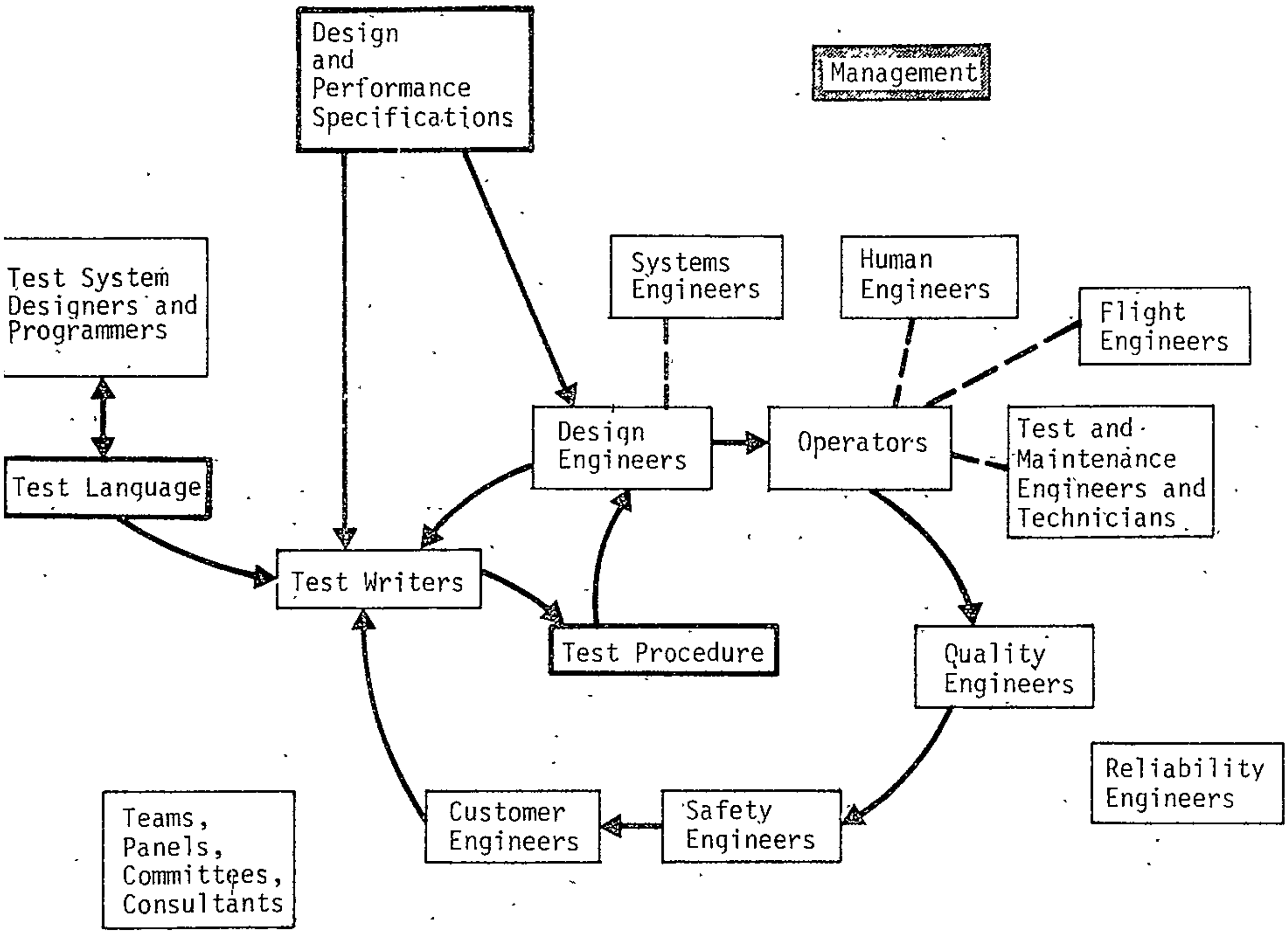


Figure 5-1 Test Programming Activities Flow

engineering and test activities.

A second objective should be to keep it somewhat independent of internal working of the test system, i.e., allow the language to specify functions as test system/prime system/operator interfaces rather than specify test system internal actions.

When the test writer becomes proficient in the use of the programming language, he generally desires means of "abbreviation" of the language elements, statements, and routines in order to save writing time and inadvertent errors. The language should allow the writer to predefine terms and abbreviations which can sebsequently be used to shorten the writing task. The defined items can be restored by the language compiler-processor in any printouts.

By means of a keyboard, the test may be input to any of several recording mediums (punched card, punched tape, mag tape, disk, etc.). A printout of the test procedure in the source language is made by the compiler-processor for review by the writer and many other involved users.

5.2 Design Engineer

Depending on the particular organization, the test writer may or may not be a designer or representative of the system or unit being tested or controlled. If he is not, it is generally considered necessary that the test programs be reviewed and approved by design engineering personnel.

The design engineers include a broad range of disciplines such as electronics (including digital systems), hydraulics, pneumatics, propulsion, RF systems, and life support. In each case, however, the designers are accustomed to the use of electrical signals for control and monitoring. It might be possible to restrict language elements to the definitions and descriptions that would exist at actuator and transducer electrical interfaces, however, this is not believed desirable or necessary, because the relationships between the non-electrical parameters can, in general, be uniquely and completely defined in terms of the electrical characteristics. No real language barrier exists.

One language requirement is that it consist of familiar technical terms and a logical, readily understood format. The prime equipment design engineer should not be required to learn details of the test system and its internal operations in order to understand the test and control interfaces with his equipment. In-so-far as practical, the language should be test system independent.

5.3 Operating Personnel

The eventual execution of the program will involve test and control

operators. They must be intimately familiar with all operator interface hardware and with the system level operating characteristics of all hardware elements of the prime equipment and the test (or control) system. They are customarily test engineers at higher equipment operation and test levels and advanced technicians at lower (LRU) test levels. In the ultimate system operation level, they may be flight engineers, astronauts, or pilots. If on-line programming capability is included in the concept, they will be proficient in use of the language or a special subset. In any case, these language users will review and approve the test procedure. Of prime concern will be the human engineering aspects of operator interactions, instructions, decisions, holds, emergency routines, displays, etc., and any related aids that will be made available. The language requirement for this user is primarily that it be easily read and understood and that operator involvement can be clearly and non-ambiguously defined.

5.4 Quality Assurance

The quality assurance personnel are involved in several areas of program preparation and execution. One role is to assure that the test procedure meets or exceeds all documented test requirements and indeed verifies the required performance capabilities of the system or unit under test. Another is to verify, either during or after program execution, that the tests and/or control activities were indeed performed and that acceptable results were obtained. The language requirements of this user include non-ambiguous, English-like readability, ability to clearly state evaluations and decisions, and a means to clearly specify how the system is to display and record the results of test evaluations, significant branches, and test completion.

The task of verifying that the test and control system acceptably executes the actions directed by individual test language statements is a separate task performed perhaps by a different group of quality assurance personnel. This separate task need not be repetitively performed for each new test program. It will require programming and related skills during the development and validation of the test system and the language.

5.5 Safety Engineering

Verification that test and control procedures contain satisfactory emergency safing routines and precautions is usually assigned to a specific organizational group or panel. Due to the potentially hazardous activities involved in all phases of space programs, there are customarily many checks and rechecks of the integrity of test and control programs as well as equipment. The performance and integrity of equipment at all levels of testing is of concern, because it will eventually be used at higher levels and in hazardous operations.

The language must be easily read and understood by engineers, and facilities for clearly defining and presenting warnings, precautions, safing routines and monitoring must be provided.

5.6 Customer

The term "customer" as used herein includes all of the generally higher levels of program or project management such as those implied by headquarters, project management, integration, coordination, etc., that are normally attributes or roles of customers.

The background and training of involved personnel may span several technologies, including engineering in almost every case and programming in some cases. Here again, the programming language requirements are readability for experienced engineers without extensive training in the use of the language. It should not be necessary to acquire a detailed knowledge of the test or control system in order to understand the test program.

5.7 Other Users

In addition to the personnel and organizations previously identified, there are many other groups of technical people who must have some degree of familiarity with the test language and the resulting test procedures. Included in this group are managers at several levels who are responsible for the personnel that use the language. They must select and evaluate the specific users, estimate time and personnel requirements, and make decisions as to the utilization of the language.

The test equipment designers and programmers who design hardware and machine language programs to implement the test language statement types are obviously involved, as are the compiler/translator designers. The final design of the test language can be greatly influenced by this group of specialists. As a consequence, many past "test oriented" languages are in effect test system programming languages (anologous to machine language) that direct the internal communications and hardware activities of the test system rather than direct the control and evaluation interfaces for the UUT. When this is the case, it is frequent practice for the real test designer to provide a separate, detailed test procedure document in English and other forms as input to the test system programmer. The programmer then does the (manual) translation to the test system language. The other users will then require more training.

The primary (and uncompromising) requirements placed on the test language by this group is that each and every implication of the language be precisely and non-ambiguously understandable in terms of test system actions, so that compilers, translators, and/or interpreters can be designed. Obviously, the test system and computer programmers are intimately involved in the design of the

test language, because they are responsible for implementing it.

5.8 User Related Considerations

As the previous discussions imply, an English-like, engineering oriented language can best serve the requirements of the variety of users. Such a language should require minimum training and would provide a precise self-documenting language thereby obviating separate test specifications.

Not many years ago, test programs for computerized test systems could be prepared only by professional programmers. The test requirements input and the computer program devlopment had to be separately and formally organized and documented. The test engineers and computer programmers had difficulty in communicating. Resulting computer programs were not intelligible to the test engineer, much less the host of other responsible personnel. Computer controlled, fully automatic testing was resisted because of this aura of mystery surrounding the computer and programming. Without a full understanding of the systems and programs, problems were attacked somewhat in the same manner as New England once attacked witches.

As computerized systems became more widely accepted, techniques for breaking the barrier that existed between the users and the programmers were developed. One of these techniques has been the evolution of higher level test oriented languages. It provides the user with an understandable tool for describing the tests clearly and completely. The necessity for a common language that both the variety of users can understand and that the computerized system can directly accept as a program eliminates the communication barrier represented by manual translation to lower level languages.

5.9 Language Related Considerations

One of the major considerations influencing the test language requirements is the degree of integration of checkout with normal "operational" functions. The application of Built-in-Test Equipment is a hardware version of this philosophy, and computer self-check is more nearly a software version.

The following concepts are all expected to be used in the Space Shuttle and other future space systems, and will all impact the distinctions between "test" and "operation":

Built-In-Test-Equipment (BITE)	Majority Voting
Built-In-Test	Dissenting Vote Indication
Self-Test	Transmission Verification
Automatic Redundancy Channel Selection	Parity Checking

The implementation of these functions may be decentralized, i.e., incorporated in LRU's and subsystems, or centralized in varying degrees. The objective of automatic safing and autonomy are well served by these concepts and they will undoubtedly continue to evolve throughout the design and development phases of all future programs.

The "test" orientation of a programming language requires a definition and distinction of testing functions for autonomous systems. This definition will not be available for any specific system or subsystem until its design is essentially complete and static.

Most of the discussion herein assumes that the "test system" is distinguishable from the system-under-test. This is not always the case. Built-in test capability, either in the hardware or in a data management and control system, may result in the loss of the functional boundaries. It is frequently true that operations related to test and monitoring are functionally distinquishable from those required for the prime equipment to perform its basic mission. The very concept of monitoring, however, may be integral with the operational concept when it is a basis for control actions for the operating system. This is true for launch control and it is true for redundant systems mode selections.

6.0 LANGUAGE OBJECTIVES

A test oriented checkout language, designed to meet Space Shuttle requirements, should meet the following Objectives:

6.1 Independence With Respect To Testing Equipment

The conceptual nature of the Space Shuttle systems at this time makes it highly desireable that the test language be developed independent of any testing equipment. There are expected to be several differently configured test systems which should use the language during the operational phases; there will also be numerous additional test system configurations at contractor and vendor facilities.

6.2 Flexibility

The Space Shuttle is an evolving concept and is expected to change significantly in the near term. As a result, the test language must provide flexibility to meet these expected but as yet undefined changes.

6.3 Engineering Reader Orientation

Two approaches to the definition of a language with respect to the users of that language can be identified. One approach is to define a language with maximum ease of writing (which generally results in degraded readability). Another approach is to define a language with maximum readability (which puts a heavier burden on the writer).

In space vehicle checkout applications it has been historically true that the writing task is a relatively smaller portion of the overall programming cycle, while the resulting tests must be read and validated by a number of people. Therefore, the emphasis should be placed on maximizing readability and providing aids within the language to assist a reader in understanding tests written in the language.

6.4 Concurrent Test Execution Language Capability.

Concurrent test execution is necessary due to the complexity of the Space Shuttle itself and the requirement to check out the subsystems of the shuttle in a relatively short time. Multiple data buses are attached to each computer in the Space Shuttle central computer complex. This allows a concurrent test execution capability in which multiple tests can be carried out at the same time and on different buses, thus speeding up the overall check out cycle.

6.5 Self-extension Capability

A self-extension capability is necessary to enable the language

to keep up with new developments in space vehicle checkout without resorting to language and compiler modification which is a time consuming process requiring professional programmers. The nature of the Space Shuttle program in the near term requires that this flexibility be available in the language to make it responsive to changes which cannot be anticipated at the present moment. An important consideration is the constraint of such a capability so that difficulties are not introduced for those who must read and interpret the resulting language extensions.

6.6 Computer/Computer and Computer/Digital Interface Unit Communication Capabilities

Present Space Shuttle concepts require multiple computer configurations in a central computer complex linked by multiple data buses to other computers and special digital interface units. These computers and special digital interface units in turn interface with the line replaceable units. Test and checkout of the installed line replaceable units requires communication between the computers and the digital interface units via the data buses. Other communication lines are presently envisioned for the Space Shuttle ground support systems.

6.7 Maximum Use of Past Language Development Efforts

It is anticipated that a number of languages may be involved in the Space Shuttle program. At a minimum a language will be available for programmer use in the development of software systems and application packages and a test oriented language will be available for use by test engineers. To increase understanding and the ability to communicate, attention will be paid to features of those languages which are likely to be used on the Space Shuttle program in conjunction with the test oriented language being defined in this study. Attention will also be paid to those languages which have been developed and used in the past and are already familiar to those working in the spacecraft field.

The Abbreviated Test Language for Avionics Systems (ATLAS) is a prime language for consideration in this respect for several reasons. As a result of the Phase I effort of this study it was determined that the ATLAS language provides many of the desirable characteristics which are felt to be necessary for a test and flight engineering oriented computer language.

Of the languages studied, ATLAS is the most readable with regard to a widely varing group of users. It has been widely accepted in the commercial airline field and among aircraft and aircraft equipment (LRU) manufacturers. It has potential for long term use as it has been designed to be independent of particular test equipment and provides for test and checkout of a large number of different equipment types. It is the least programmer oriented language of those studied.

7.0 LANGUAGE CHARACTERISTICS

A test oriented checkout language designed to meet Space Shuttle requirements should have the following characteristics in order to achieve the objectives defined previously:

7.1 Test Orientation

The following discussion will identify the test oriented functions which need to be implemented in the Test and Flight Engineering Oriented Language.

One form of implementation of the test oriented function which has been successfully used in previous ATLAS compilers and is regarded favorably for use in the new language is the concept of a core subset. A core subset of a language is a basic set of action primitives from which all other required action primitives can be defined. This definition can be done at the time the language is designed or be available as a capability for use by the test writer himself. The language characteristics to be described below will provide this capability to the test writer as well as defining certain general action oriented functions which will be based on a core subset.

This approach provides for simplicity in language definition and compiler development along with great power in terms of availability of the language for use in a very wide test application area. This power can be used by the very sophisticated test engineer to assist him in the performance of his function. The less sophisticated test engineer does not suffer in the performance of his function as the basic language provides all the necessary capabilities. In either case the functions of the readers and reviewers of this language is not compromised as all information will explicit in the resulting source listings.

7.1.1 General Nature of Testing.

Operation of the Space Shuttle consists of

- making logical decisions
- performing the necessary control actions
- monitoring results
- correcting the responses

After manual initiation of a control, the automated system performs the desired activity. Testing involves the initiation of activity with controlled predetermined conditions and then analysis of the resulting activity. The predetermined conditions are in the nature of applied stimuli while analysis involves the mearuring and comparing of the responses. It is this activity and analysis that the

language of Space Shuttle must concern itself.

The action words become the verbs in the Test and Flight Engineer Oriented Language. This section will devote itself to describing test activity. The exact nature and selection of verbs is deferred to Phase III of this study.

7.1.2 Initiation of Test Execution Via The Language

To initiate the action of a test, the language must be able to call or perform a test sequence. Such a request for action permits a test to commence.

The test being called into execution would have previously been compiled on an off-line system.

It is to be noted that the test and checkout system executive should provide a non-language capability to allow an operator to use the keyboard or other control media to initiate the execution of required tests. The language will be independent of the test system.

7.1.3 Application of Stimulus

The first test function usually performed is characterized by such terms as; apply, stimulate, set, or turn on. These signify the application of a specific stimulus or control signal to a specific unit under test.

Application of stimulus in the case of shuttle requires a command to be issued (digital) from the controlling computer to the desired IU. The digital command work will be translated by the IU and will respond with a stimulus signal to the addressed LRU.

The application of stimulus signals may take many forms. Major categories include DC signals and AC signals, normally classified as analog signals. Application of single level D.C. signals usually falls into the discrete category. A third category consists of digital stimulus. The nature of the shuttle (with its integrated avionics) indicates that a built-in stimulus (contained in the IU) will have to be programmed. As far as the test writer is concerned, he must request the application of the stimulus just as he would in any other test situation. Where the natural or operating stimulus cannot be called into use, an artificial stimulus is applied which produces a known output for a known input.

7.1.4 Measurement of Output Signal

Once a stiumlus is applied to a unit under test, an output is expected in response to the input stimulus. The language must provide for acquiring that output and retaining it for further manipulations. In the Space Shuttle application, outputs will be sensed by IU's attached to LRU's and the data then is placed on the

data bus to be received by the central computer complex. This function is generally characterized by such terms as; measure, monitor, test, or sample.

7.1.5 Comparison of Results

It is generally necessary to determine if the output acquired as a result of a measurement function is satisfactory with respect to some expected value. This output value is then compared to some predetermined value, with appropriate tolerances; and the results are used to indicate some further action. This function is generally characterized by such terms as; compare or check.

The further actions that a test may take are provided for in the characteristics described in the rest of this section.

7.2 Naturalness of Statement Structure

The statement structure should be engineering oriented English format with minimum use of abbreviations and identifier codes.

The English-like format of the language will enhance the capability of a varied class of readers to understand the tests written in the language. The potential for error on the part of the test writer will be reduced due to the familiar and natural way of using the language. Ease of learning on the part of all users will be enhanced by an English-like format.

7.3 Self-extension Capability

In accordance with objective 6.4, self-extension capability should be implemented in the language. This self-extension capability should be primarily provided for the use of the sophisticated test programmer who takes the time required to study how the language may provide powerful assistance in the accomplishment of his particular task. It is not intended that this capability be used by the less sophisticated test writer and in no way should detract from his ability to use the more straight forward portions of the language. Some project control of the use of language extension capabilities may be desirable.

7.3.1 Macro Definition Capability

A macro definition capability should be implemented in the language to allow new primitives to be defined via the use of a basic primitive set (core set).

A macro definition capability allows a writer to tailor a basic primitive set to his particular testing requirements. New complex primitives may be built up in this way from simpler more basic primitives. Reader understanding would not be compromised as the complete macro definition can be part of any resulting

source program listing.

7.3.2 Other Languages

A capability should be implemented in the language to allow the sophisticated test programmer to include in his code the use of other languages.

Other languages will be used in the Space Shuttle program in conjunction with the Test and Flight Engineering Oriented Language. Such languages may be; CLASP or one of its derivatives for guidance and navigation, possibly ATLAS for LRU checking in a ground support facility, and one or more machine languages. A capability to leave the Test and Flight Engineering Oriented Language for a period of time to operate in one of these other languages may prove to be of use.

7.4 Self-Documenting Capability

7.4.1 Language Primitives

The language primitives should in themselves provide self-documenting capability.

7.4.2 Comments

Comments should be allowed in any statement where multiple blanks may appear. The use of comments in this way will allow the writer to clarify any statement that is not completely clear as a result of its primitives.

7.4.3 Define-Type Capability

A define-type capability should be provided as a writing aid. In essence, this capability provides a writer with the ability to create within the language a set of abbreviations (identifiers) for primitives and combinations of primitives and statements. The define type statement will help the writer to both minimize the possibility of error in repeating long strings of primitives and will also ease the writing task. The task of the reader is not compromised however, since the compiler will produce full listings with proper substitutions for all abbreviated portions of statements.

7.5 Safing Features

A capability within the language should be provided which allows the test writer to create his own safing features.

Three approaches with respect to safing features can be identified. One is the inclusion in the language of the necessary capabilities to enable a test writer to create his own safing procedures which would be attached to the test he is currently writing versus the

inclusion of a standard set of safing procedures either in the language itself or as part of an operating system. Inclusion of a standard set of safing procedures in either the language or an operating system is difficult to do prior to the establishment of the actual operating hardware of the checkout system. Since it is desired to make the language independent of any particular test system, it will be necessary to provide to the test writer the capability to create his own safing procedures. Another advantage to this approach is that safing procedures can easily be modified when necessary by the creation of a new procdure.

Safing procedures might be called into execution by the operator, by branches within a program, or by interrupts (which are discussed later).

7.6 User Program Maintenance

User program maintenance will be facilitated by the naturalness of statement structure and the self-documentation capability of the language.

This function is generally not the responsibility of the test writer but the responsibility of the users of the tests. In any case any changes which are initiated to a test are subject to considerable review by a number of affected parties.. This requires that such changes and the test itself be readily understood by all concerned. The engineering reader orientation and the self-documenting capability of the language are of primary assistance in this capacity.

7.7 General Charactistics of Language Processor

The following characteristics are proposed for a Test and Flight Engineering Oriented Language processor.

7.7.1 Off-line Processing

The central computer (quadruply redundant) in the Space Shuttle is currently baselined at 64 K core. Most of this core is presently assigned to the multitude of functions which the central computer must carry out. Since a language processor for the Test and Flight Engineering Oriented Language will be a large and complex program, the necessity for off-line processing is apparent.

7.7.2 Extensive Error Checking And Diagnostic Capability

A limited checking capability in the language processor will place more of a burden on the test verification activity. Since the test verification activity is important and time consuming, any assistance that can be given via the language processor is worth the extra effort necessary to build in error checking and diagnostic capabilities.

7.7.3 Translator-Interpreter Approach for Code Generation for Several Target Machines.

Space Shuttle contains many computers attached to different subsystems, each having some responsibility for the checkout of its subsystem. A way of generating code for these, possibly different, computers is necessary.

Two approaches to creating a test oriented higher level language processor can be identified. One is a translator-interpreter approach and the other is a compiler approach.

A translator processes sets of input source statements, reducing them to sets of data words for storage until needed for execution. An interpreter then accesses the stored data words and uses them as information which directs the execution of routines designed to accomplish the required testing.

In order to generate code for the various target machines, a compiler would need a set of code generating routines for each target machine. A translator approach creates a test sequence data list which is identical for all target machines. Interpreter systems are then built for each target machine within the shuttle configuration using as a baseline a generalized interpreter as developed for the main shuttle computers.

Core usage for the resulting tests must also be considered. Appendix A compares the code for tests generated by the OCS TOOL language against a hypothetical TOOL language compiler. The results indicate that a translator-interpreter approach provides an overall core savings where more than minimal testing is involved.

7.7.4 Validation of Test Operations With Respect to Test System and Shuttle Requirements.

This function is recommeded for the same reasons as indicated in paragraph 7.7.2.

7.7.5 Language Processor to be Written in a Higher Level Language

Approaches which can be identified here are the writing of the language processor in machine language versus the writing of the language processor in a higher level language. Significant advantages are available if the language processor is written in a higher level language.

- More rapid development of the language processor.
- Easier communication between developers of the language processor via the higher level language representation.

- Considerable increase in the ease of modification of the language processor.
- Ease of training of those who are to maintain the language processor.
- Simplification of the required final documentation.

7.8 Format

7.8.1 Statement format should be free form with respect to input media.

Statements may be fixed, partially fixed, or free form. A free form type of format provides the following advantages with respect to readability.

- The meaning of primitives depends solely on their alph-numeric configuration and not on any specific orientation with respect to input media. Neither the writer or the reader need be required to recognize meaning based on the position of a primitive. All meaning is therefore explicit in the statement.
- If any specific fixed or partially fixed format is needed for any reason, it can be accommodated easily with no change to compiler or language. Complete flexibility is provided with the free form format.

7.8.2 Primitives Should be Ordered in a Natural English Manner

Primitives may be ordered in an arbitrary way, completely unordered or ordered in a natural English manner. It is felt that the first two ordering schemes will result in difficulty for the reader of a test. The readers will be most comfortable with statements that appear in as natural a form as possible. The writer is also prone to error when he is required to write in an arbitrary format or an unstructured format. The latter, especially, can be prone to inadvertant omissions.

7.8.3 The End of a Statement Should be Signified by a Period.

The end of a statement can be specified via a symbol or via the start of another line on an input medium. The requirement of signifying the end of a statement by a symbol frees the statement format from any dependency whatever on the input media. A compiler written for the language will, therefore, accept source data from any available input medium. The use of a period makes the statement more English-like.

7.9 Character Set

LETTERS: A thru Z

NUMBERS: 0 thru 9

SYMBOLS: + - * / ( ) . , _ ' = ?

This character set is a subset of the basic set common to both the EBCDIC and ASCII code formats as shown in IBM System /360 Principles of Operation, Form A-22-6821-6. It should, therefore, be a set available in any computer input equipment.

7.10 Significance of Blanks

Blanks should be used to delimit numbers and primitives. Successive blanks should not be significant.

Numbers and primitives can be delimited either by using special symbols or blanks. In order to further the English-like character of the language and for ease of writing, blanks should be used as delimiters.

7.11 Comments

Comments should be inserted at any place where multiple blanks are allowed. Comments should be delimited by double apostrophes at the beginning and the end.

Since the language is to be independent of any particular input media, comments can be inserted either anywhere multiple blanks appear or only between statements. It is felt that full flexibility here will enhance clarity in any statement that is not completely clear as a result of its primitives.

7.12 Operators

7.12.1 Numeric Operators

- Addition
- Subtraction
- Negation
- Multiplication
- Division
- Exponentiation

The lack of an arithmetic calculation capability was identified in the Phase I report as a deficiency in some of the test and checkout languages studied. In order to avoid this deficiency in the new language a capability identical to that in the ATLAS language should be provided.

7.12.2 Relational Operators

The following relational operators should be provided

- Equal
- Not Equal
- Greater than
- Greater than or equal to
- Less than
- Less than or equal to

These relational operators are necessary to aid in expression of the various conditional statements, limit checks, and other forms of checks universally required in test and checkout languages.

7.12.3 Logical or Boolean Type Operators

No immediate need is apparent with regard to the Space Shuttle test and checkout application for general purpose logical or boolean type operators. This is consistent with present ATLAS language usage.

7.13 Primitive Terms

Language primitives should be English-like words.

Language primitives can be coded mnemonics, abbreviated forms, or English-like words. The use of coded mnemonics or abbreviated forms, while of assistance in writing in the language, places a burden on those who must review the resulting tests. They are required to know the language in considerable detail to be able to fully understand the tests that are being reviewed. An English-like format will enable the reviewers to more readily grasp the intent of the tests and will require less training in its use.

The define type capability mentioned in 7.4.3 enables the test writer to take advantage of all the coded mnemonics and abbreviations he cares to, while preserving the English-like format in the final source listings.

7.14 Delimiters

- Language delimiters should be natural English forms.
- Statements should end with a period.
- Comments should be enclosed in double apostrophes (quotes).
- Data names should be delimited with underscores.
- Primitives should be delimited with blanks.

Language delimiters can be arbitrary special symbols or natural English symbol usage. For an engineering reader oriented language, natural English symbol usage is preferable. The use of underscores to delimit data names is a deviation from English like usage which is justified on the grounds that a clear identification of data names in a statement is a desirable feature for reader understanding. Use of an underscore is an unobtrusive way to achieve this identification.

7.15 Identifiers

7.15.1 Statement Labels

Statement labels should be of the form:

STATEMENT N where N is a one to six digit number.

Statement labels maybe arbitrary names, numbers only, or a restricted form such as the one chosen. Arbitrary names for statement labels would require a special delimiter to distinguish a statement label from a data name. This would require the users to learn a non-English like primitive. Simple numbers as statement labels are readily distinguishable by the position they occupy with respect to the rest of the statement, but again a non-English like form is being used. The requirement of labeling a statement by the form STATEMENT N is felt to be the clearest approach with respect to the user while maintaining an English-like format.

7.15.2 Data Names

Arbitrary data names should be alphanumeric character strings of up to 32 characters.

The symbol underscore _ should be used to delimit a data name.

An alternative delimiter for data names is the single apostrophe as is used in the ATLAS language.

Data names may be completely defined within the language or may be arbitrary with a specified number of characters. Since Space

Shuttle is an evolving concept and is expected to change in the near term, a fixed set of data names is not feasible at this time or any time in the near future. Allowing arbitrary data names of up to 32 characters gives the test writer the flexibility needed to use full English words for meaningful data names. Data names are delimited to clearly identify these items in a language statement (See paragraph 7.14).

7.16 Arrays, Lists, and Structures

7.16.1 Arrays

A single dimensional array capability should be provided.

This capability provides for the grouping of data items with the same characteristics into an entity referenced by a single data name whose individual items are distinguished by an index value. This type of data grouping is useful in many operations involving repetitive usage of similar data items. The latest version of the ATLAS language will implement such a capability.

7.16.2 Lists

No list processing capability will be implemented.

No immediate need for a list processing capability is apparent with respect to the Space Shuttle test and checkout application. This capability is a highly sophisticated programmer oriented capability and therefore, would not be appropriate for the use of test engineers.

7.16.3 Structures

A structure capability should be provided.

Data structures, or tables, are data groupings in which differing data types are associated together and this association is given a unique data name. Applications of this form of data grouping are readily apparent in the Space Shuttle test and checkout environment. Comparisons of differing data types representing the functioning of a device can be easily facilitated with this capability. Data dictionaries will have to be developed (See 7.17) which can conveniently be expressed using table structures. This is essentially a convenience capability which also enhances readability by associating dissimilar data with common functions.

7.17. Dictionary Data Banks

A dictionary data bank capability should be available in the language to provide the LRU designers and the test equipment designers with the capability to declare the nouns and modifiers required to test a unit and to define the action of those nouns and modifiers with respect to the test system.

This requirement is necessary to provide the final link between the language and the test system. Such a link must be supplied in one way or another. The alternative to creating a language capability to define that link is to have a programmer generate machine language tables which provide the necessary information. These tables could be included in the language processor or operating system at the time the unit and test equipment have been designed. To avoid the use of a professional programmer to modify the language processor each time new LRU's (requiring new nouns and modifiers) and new test equipment are available for use, a language capability is recommended.

This language capability provides for complete test system independence of both language and language processor. It will provide the capability required to interface tests written in the language with any test system.

A hierarchy of language users is necessary under the data dictionary concept. The LRU designer specifies the nouns and modifiers which are required to completely implement the test functions available in the language. The test equipment designer specifies the meaning of these nouns and modifiers with respect to the equipment which will actually test the device. In the case of Space Shuttle, for instance, a noun signifying pressure would have to be defined in terms of IU numbers and digital code words. This information in placed in a dictionary data bank, utilizing special language capabilities designed for this function.

When the test engineer writes his test he uses the functions available in the language along with the particular dictionary data bank he needs to provide him with all allowable nouns and modifiers which can be used in testing the particular device in which he is interested. He is in no way concerned with how the test system implements the meaning of these nouns and modifiers.

7.18 Program Structure

The structure of the source listing for a test written in the Test and Flight Engineer Oriented Language should be as follows:

- Dictionary data bank identifier.
- Dictionary data bank noun and modifier information (supplied by compiler).
- Macro definitions of new primitives
- Define statements establishing abbreviations and identifiers for use by the test writer.
- Test writer defined subroutines.

- Library subroutines (supplied by the compiler).

- Main body of the program.

This program structure should be selected to provide to the user of the test all data and information which is needed for proper understanding of the test. This information is presented in a logical manner with all required data and program sub-structures presented before the main program.

7.19 Block Structures

No specific block structure capability appears necessary for the new language beyond that provided via the subroutine capability (7.21). It is felt that to provide the test engineer with too much in the way of this type of essentially programing oriented capability would be confusing and not in keeping with the objectives of the language.

7.20 Loop Structures

Loop structure capabilities should be provided in the language.

It is anticipated that some looping capability will be implied by specific test and checkout functions and these should appear in specific primitives in the language. A capability for looping independent of a specific primitive should be provided to allow the sophisticated test engineer the capability to include loops in any new primitives he may create via the self-extension capability of the language.

7.21 Subroutine Structures

A subroutine capability should be provided in the language.

This capability is a powerful aid for specifying those functions which are repeated many times. It is both a convenience to the writer in reducing his writing task and assists the reader by isolating and clearly specifying those functions which are of a repeatable nature.

This is a programming oriented capability provided for the use of test engineers. All test oriented languages studied in Phase I have some form of subroutine capability.

7.22 Library Capability

A library capability is regarded as necessary for the Space Shuttle test and checkout system due to the large number of individual

units which must be tested in each subsystem. Subsystem tests will be made up of a large number of these individual tests. However, this capability is usually provided as a portion of an operating system and as such no reflection of this capability appears in the language.

7.23 Interaction with the Operative System

No interaction with an operating system should be specified in the language.

In conformity to the requirement of a language independent of a particular test equipment configuration the language will contain no facilities for interaction with an operating system.

7.24 Data Types

Investigation of the Space Shuttle test and checkout applicationa and previous efforts at the definition of test languages leads to the conclusion that the following constant and data types are required in the new language.

7.24.1 Constants

- Integer
- Fixed Point
- Boolean
- Text
- Binary
- Time

7.24.2 Data Variables

- Integer
- Fixed Point
- Boolean
- Text
- Time

7.25 Formula Types

Consistent with the operators and data types previously specified,

the following formula types are required to fully utilize these operators and data types for the Space Shuttle test and checkout application.

- Numeric formulas
- Relational formulas

No text manipulation formulas are necessary as text should be provided for output of predefined text messages only. Data values within messages should be capable of being modified but the text should remain fixed.

7.26 Assignment statements

Consistent with the requirements for operators, data types, and formula types; the following types of assignment statements are required.

- Simple numeric assignment statements.
- Simple Boolean assignment statements.

No text type assignment statements are necessary as text constants should be assigned at the time a text data variable is declared. This is consistant with the limited usage made in the language of text strings.

7.27 Sequence Control

Statements should be executed in the sequence in which they are written except as altered by unconditional or conditional transfers.

- A simple GOTO primitive should be provided for unconditional transfers.
- A statement of the form IF, THEN should be provided for conditional transfers.
- A repeat statement should be provided to enable the test writer to repeat a previously written statement.

These capabilities for sequence control appear in one form or another in almost all the test oriented languages studied in Phase I Only the simplest forms of sequence control should be required in the language in keeping with the objective of an English-like language. More complicated forms of sequence control are oriented toward programmer usage.

7.28 Interrupt Initiated Routines

A capability should be provided for the processing of interrupt

initiated routines.

7.28.1 Inhibit/Enable

An inhibit/enable interrupt capability should be provided.

This capability allows the test writer to control the action of those interrupts which affect the operations of his test.

7.28.2 Execute Test on Interrupt

The capability to specify a test to execute on the occurrance of an interrupt should be provided.

This capability provides a test writer with the ability to respond to an interrupt which may affect the operation of his test. These interrupts may be interrupts that specify that certain error conditions have been generated in the hardware device under test, over and above those conditions which can be determined in the normal course of testing.

This capability of processing interrupt initiated routines can be modeled after that same capability as provided in CLASP or STOL.

7.29 Compiler Directives

Compiler directives are necessary for the following functions which have been previously described.

- Macro definitions.
- Other language use capability.
- Define capability.
- Dictionary data bank capability.

7.30 Man/Machine Interface

A display and an input/output capability should be provided in the language.

The Space Shuttle cockpit as presently configured provides CRT displays, microfilm displays, in-line alphanumeric displays, light driven warning indicators, and a printer. The ground Control Center will probably duplicate these items. The use of many of these output devices will be necessary to output the results of the tests to the test or flight engineer. Language capabilities are, therefore, necessary to drive these devices.

Control information to the system will originate from switches and from the keyboard. A language capability to accept this input

information is necessary.

7.31 Records and Logs, Time Tags

A record capability should be provided in the language.

The Space Shuttle has a maintenance recorder on board for the purpose of recording information regarding the operation of sub-systems and line replacable units. This information will then be examined at the conclusion of a mission to determine which devices may require service. The test and checkout system will provide some of the inputs to the maintenance recorder. A language capability to provide for these time-tagged inputs is, therefore, necessary.

7.32 Multiple/Parallel Actions

A concurrent testing capability should be provided in accordance with objective 6.3.

A special set of primitives to facilitate concurrent testing, along with simple rules for their use, should be designed into the language. Such multiple programming features do not overly complicate the language or its compiler but provisions for concurrent testing must be included in the executive programs. Resource allocation provisions in the executive program assure that two or more programs being executed simultaneously will not adversely interact.

The language should contain a statement similar to

```
        PERFORM_PRIME_
              :
CONCURRENTLY PERFORM_ALPHA_
```

This would cause the executive program, operating at RUN time, to start program ALPHA and execute it simultaneously with program PRIME. Program ALPHA could also contain a statement within it:

```
        PERFORM_ALPHA_
              :
              :
CONCURRENTLY PERFORM_BETA_
```

Thus program BETA would be started at that time and would be executed along with PRIME and ALPHA. If it is desirable to synchronize, for example, BETA with PRIME; program PRIME would contain a statement.

```
        PERFORM_PRIME
              :
              :
CONCURRENTLY PERFORM_ALPHA_
              :
              :
       SYNCHRONIZE  n
```

At this time PRIME would hold until program BETA reached the statement.

SYNCHRONIZE n

At this time both programs would continue concurrently. If BETA reached

SYNCHRONIZE n

before PRIME reached

SYCHRONIZE n

BETA would then wait for PRIME.

Thus multiple programs can be run concurrently and synchronized where desireable.

7.33 Monitoring

A language capability should be provided to enable a check to be utilized in a continuous monitor mode.

This capability is necessary to allow the monitoring of the Space Shuttle systems continuously. As long as no anomalies occur, little notice is attached to the monitored systems. However, if an anomaly is detected, a previously defined warning, alternate action or a backout routine provides corrective action.

While pressure is being applied to Space Shuttle pressurization spheres or while propellants are being loaded, possible hazardous conditions should be constantly monitored until a specified pressure is reached or exceeded or until a specified temperature is reached or exceeded.

With the capability for concurrent test execution (See 7.32) existing in the language, monitor tests can be continuously executed while other tests are run on a noncontinuous basis.

A monitor test differs from a normal test only in that a way of specifying repeatable execution exists for the monitor test.

7.34 Test System Dependency

No interaction with a specific test system should be specified in the language.

This requirement is in conformity to the objective of a language independent with respect to a particular test equipment configuration.

7.35 Clock and Time Controlled Actions

The capability for the following clock and time controlled actions should be provided in the language.

- Delay for time interval.
- Delay until a specific time value is reached.
- Optional execution rate control.
- Initiate event at time specified.
- Acquire external time.
- Initialize internal time.

Timing functions of this nature are required in the language due to the time dependent nature of the test and checkout process. Various methods of achieving these capabilities appear in most test languages.

A promising approach to implementation of the first three capabilities without the definition of separate primitives is the optional tagging of individual statements with "time to execute" indications. Both forms of delay and full control of execution rate would be available to the test writer where these functions were necessary.

7.36 Multiparameter Tests

Features are provided in the language, which have been described previously, to allow the test writer to create a multiparameter testing capability.

Each primitive in the language which specifies a particular test function involves only a single parameter. From these test function primitives the test writer through the use of the macro capability and the subroutine capability can implement any multiparameter testing capability he desires.

In keeping with the objective of engineering reader language orientation, this approach provides an explicit way of creating multiparameter tests. The reader has all the information available to him on a source listing to enable him to determine exactly what the test entails. He is not required to remember special multiparameter test function primitive configurations.

7.37 Special Discipline Provisions

Special discipline provisions within the language should be confined to words which identify special characteristics which are

attached to declared data items.

This approach removes special discipline provisions from the test function primitives which should be designed for the general testting problem. It confines these characteristics to data which represent the subsystems and LRU's under test.

It is necessary to have these special provisions due to the requirement in Space Shuttle for testing a large number of different devices and subsystems. The various users of the test language, as a result, will span many disiplines.

7.38 Interface Characteristic Specifications

A limited capability for interface characteristic specifications should be provided in the language.

The Space Shuttle hardware as presently defined is intended to have fixed interface specifications. A complete capability as found in the ATLAS language will not be necessary to accomodate the Shuttle equipment.

The interface specifications necessary will be defined via the data characteristics available in the language.

7.39 Test Level

The language should be capable of defining tests at all levels; system, subsystem, unit and sub-unit.

The use of the same language will facilitate the preparation and verification of test programs at higher levels because the writers and readers can directly use and compare performance parameters, etc. In addition, the separate programs can utilize common definitions, subroutines, and libraries when they are applicable. The subsystem test engineers can readily verify performance of the subsystem and units when involved in higher level tests. Common language processors can be used.

7.40 Program (Project) Orientation

The language as described in this report is capable of being used not only for Space Shuttle but for test and checkout of other advanced space vehicles and systems.

The language characteristics as developed in this section of the Phase II report have been developed as a result of study of previously designed test languages and a knowledge of the current Space Shuttle configuration. Attention has been paid to the general test and checkout problem and the generalized needs identified as a result have been considered in establishing the characteristics of this language. The inherent flexibility and power of the

language as currently envisioned, along with its self-extension capability, should enable it to be readily applied in test and checkout of other systems besides Space Shuttle.

8.0 CONCLUSION

Phase I of this study investigated past endeavors in the field of test oriented languages. These languages all had the same goal; to provide a high level computer language to facilitate automatic checkout. All existing languages have failed (in one way or another) to meet their objectives.

In this Phase II of the study of a Test and Flight Engineer Oriented Computer language we have identified the nature of the vehicle to be tested, defined the necessary characteristics of the language, and related the requirements of the various users.

From this work we believe a specification for the Test and Flight Engineer Oriented language can be developed. Futher we feel that by making the readability of the language its key feature and making the language independent of the test equipment, that it will be easy to use and find wide acceptance.

Phase III will be devoted to the complete definition of the language, including the vocabulary, syntax, and rules for usage and expansions.

9.0 REFERENCES

Development of a Test and Flight Engineering Oriented Computer Language;
Phase I Report
MCR-70-327
Martin Marietta Corp; August, 1970

Space Shuttle Design Data Book
Phase B Study
MDC E0189
McDonnell Douglas Corp; April, 1970

STS Software Development;
Study Task 5
E-2519
Charles Stark Draper Laboratory; July, 1970

Language Requirements Analysis;
Report
Intermetrics, Inc.; June, 1970

IBM System/360 Principles of Operation
Form A-22-6821-6.
IBM Corporation; January, 1967

APPENDIX A

COMPARISON OF TRANSLATOR-INTERPRETER IMPLEMENTATION VS COMPILER IMPLEMENTATION OF OCS TEST LANGUAGE

The Onboard Checkout System (OCS) is an on-line, interactive, multiprogrammed system developed for NASA-MSC by Martin Marietta. It is an independent, real-time, computerized system designed for verification and monitoring of experimental and developmental subsystems for various space vehicles. It was developed for the IBM 4PI-EP computer. The OCS test language is called the Test Oriented Onboard Language (TOOL).

A translator-interpreter approach was selected for the OCS in order to conserve core, since no external memory was available for the initial 4PI-EP system. A Test Sequence Data List (TSDL) containing the data necessary to drive element routines would provide this core savings balanced against a somewhat longer execution time for the elements. This longer execution time is necessary due to the unpacking of data and provision of checking facilities within the element routines to process the various element modifiers.

Compilation of the elements, rather than creation of a TSDL, was investigated with respect to core usage. The resulting element sizes were estimated based on removal of packing facilities and modifier checking. This assumes that a compiler will generate code as efficient as currently exists (which gives the compiler an advantage that may not be warranted). (See Table No. 3.) The results show the translator-interpreter approach to be superior in minimizing core usage when elements are used more than three times. If no mass storage is available, a compilation approach cannot be used as the number of elements stored in main memory is severely limited.

If mass storage is available, the compilation approach is feasible for creation of tests. The choice as to its use depends on system requirements. If many tests must be stored, the translator-interpreter approach remains superior.

The following tables show the TSDL element sizes and element routine sizes within the existing OCS. Other information about the test execution system is given. This information has been derived from the OCS listings as of March, 1970.

APPENDIX A, (Cont)

TEST EXECUTION SYSTEM,
ONBOARD CHECKOUT SYSTEM.

Basic multiprogramming executive	=	536 words,	2% of core
Utilities	=	2707 words,	11% of core
Test execution supervisor and element routines	=	3858 words,	16% of core
Total	=	7101 words,	29% of core

Note: The rest of the OCS consists of the on-line-translator and the test sequence data list.

TABLE NO. 1

APPENDIX A, (Cont)

ELEMENT SIZES AS EXIST IN OCS, GIVEN IN BYTES.

ELEMENT	TSDL	ROUTINE
Begin	12-22	354
End	2	1120
Call	12	232
GOTO	4	16
Prefix	2	(Imbedded in Controller)
DO	2	34
Again	4	32
Connect	8 (Includes Delay)	10
Disconnect	8 (Includes Delay)	414
Measure	4-8	838
Stimulate	2-8	1144
Evaluate	4-16	468
Check	4-12	480
If	4	70
Delay	4	438
Display	2-28	1070
Clear	2	154
Repeat Flag	2	14
Milestone	4-12	102
Start	10	256
Stop	10	952

TOTAL IN WORDS = 2050*

*Does not include 1022 words of general subroutines referenced by the element routines.

TABLE NO. 2

APPENDIX A, (Cont)

SELECTED ELEMENT SIZES BASED ON REMOVAL OF PACKING FACILITIES AND MODIFIER CHECKING.

ELEMENT	MODIFIERS	EXISTING SIZE (Routine + TSDL Min)	COMPILED SIZE (Sizes in BYTES)
CHECK	Type, Value Data Cell, Limits	480 + 12	190
CALL	Name	232 + 12	206
DO	Repeat Count	34 + 2	14
EVALUATE	Results, Operand 1, Operator, Operand 2	468 + 4	182
IF	Flags	70 + 4	46
MILESTONE	Message	102 + 4	77

Size comparisons indicate that when any element is repeated three times or more in any single test, the translator-interpreter approach becomes more efficient in terms of core usage.

TABLE NO. 3